GROUP EXHIBIT D

XMLRecordException.java

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

/**
 * An exception that gets thrown when various XML related problems occur.
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 *
 */

public class XMLRecordException extends Exception {
   public XMLRecordException(String msg) {
      super(msg);
   }
}
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

// import java classes
import java.io.*;
import java.util.*;

// import Bridge classes
import com.ibm.xmlbridge.Syslog;

/**
 * This class ....
 * 8 is Syslog.DEBUG (whatever debug information you want)
 * 7 is Syslog.INFO (a notable event, like a ticket was received)
 * 6 is Syslog.NOTICE (something looks suspicious)
 * 5 is Syslog.WARNING (say a ticket could not be delivered on the first try)
 * 4 is Syslog.ERR (a ticket failed to be handled)
 * 3 is Syslog.CRIT (the system can no longer process tickets, say from a nearly
full disk)
 * 2 is Syslog.ALERT (page someone because the system does not work)
 * 1 is Syslog.EMERG (NT is crashing, or the building is on fire.  call in an air
strike)
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

public class XMLGatewayLogger implements Syslog {

   private int level;

   public int getLevel() {
      return level;
   }

   public void log(String name, int severity, String msg) {
      System.out.println("\r\n"+(new
Date()).toString()+":"+name+":"+severity+":"+msg);
   }

   public void setLevel(int l) {
      level = l;
   }

   public Enumeration listLog() {
      return null;
   }
}
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlbridge;

import com.ibm.patml.RuleMaker;
import com.ibm.patml.RuleApplier;
import com.ibm.patml.*;
import com.ibm.xml.parser.*;
import java.io.*;

/**
 * A class to transform tickets from one XML form to another.  It
 * is little more than a wrapper to the patml code.
 *
 * @version $Revision: 1.9 $ $Date:            06:12:16 $
 * @author Kevin McCurley (mccurley@almaden.ibm.com)
 * @author Sami Rollins
 */

public class TicketTransform {
  public static final String version="$Revision: 1.9 $ $Date:            06:12:16
$";
  RuleApplier rapply;

  public TicketTransform(String filename) throws IOException {
    try {
      rapply = new RuleApplier(filename);
    } catch (PatMLInvalidRuleException e) {
      e.printStackTrace();
      throw new IOException("invalid rules file "+filename+":"+e.toString());
    }
  }

  public TXDocument transformTicket(TXDocument before) throws Exception {
    try {
      StringWriter sw = new StringWriter();
      before.printWithFormat(sw);
      StringBufferInputStream sbis = new StringBufferInputStream(sw.toString());
      TXDocument tdoc = (TXDocument) rapply.applyRules(sbis);
      // incredible hack here to spit it into a string and reparse it.  If
      // only the TXDocument class was better documented, I might be able to
      // figure out how it should look.
      sw = new StringWriter();
      TicketErrorHandler errors = new TicketErrorHandler();
      tdoc.printWithFormat(sw);
      Parser parser = new Parser("TicketTransformParser",errors,null);

      TXDocument tx2 = parser.readStream(new
StringBufferInputStream(sw.toString()));
      if(parser.getNumberOfErrors()>0) {
        throw new TicketException("Error translating ticket:"+errors.getErrors());
      }
      return tx2;
    } catch (Exception e) {
      e.printStackTrace();
      throw(e);
    }
  }
}
```

```
/*
 * Revision history:
 * $Log: TicketTransform.java,v $
 * Revision 1.9              06:12:16  mccurley
 * reparse the TXDocument from the rules
 *
 * Revision 1.8              17:36:48  mccurley
 * static replaced by constructor
 *
 * Revision 1.7              18:47:26  mccurley
 * new Rulemaker api
 *
 * Revision 1.6              16:25:59  mccurley
 * version
 *
 * Revision 1.5              07:24:11  mccurley
 * *** empty log message ***
 *
 * Revision 1.4              21:53:21  mccurley
 * *** empty log message ***
 *
 * Revision 1.3              04:55:44  mccurley
 * *** empty log message ***
 *
 * Revision 1.2              15:27:59  mccurley
 * new package
 *
 * Revision 1.1              00:03:28  mccurley
 * Initial revision
 *
 */
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

// import java classes
import java.io.*;
import java.util.*;


/**
 * A utility class for modifying any XML in a String by removing any value-less
 * self-closing tags, like &ltTAG/&gt , with &ltTAG&gt&lt/TAG&gt
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

public class XMLTagModifier {

   /**
    * modifies the XML by removing any value-less self-closing tags,
    * like &ltTAG/&gt , with &ltTAG&gt&lt/TAG&gt
    *
    * @param inXML the incoming XML with the self closing tags
    *
    * @return the modified XML
    *
    */
   public static String modifyXMLTags(String inXML) {
      try {
         StringBuffer outXML = new StringBuffer();

         int indexOfBeginTag;
         int newIndexOfBeginTag;
         int indexOfCloseTag1;
         int indexOfCloseTag2;

         indexOfBeginTag  = inXML.indexOf("<");

         while(true) {
            indexOfCloseTag1 = inXML.indexOf(">", indexOfBeginTag);
            indexOfCloseTag2 = inXML.indexOf("/>", indexOfBeginTag);

            if((indexOfCloseTag2 >= 0) && (indexOfCloseTag2 < indexOfCloseTag1)) {
// we have a tag closing with "/>"
               // copy the original inXML to outXML till the indexOfCloseTag2
               outXML.append(inXML.substring(indexOfBeginTag, indexOfCloseTag2));

               // get the tag name
               String tagName = inXML.substring(indexOfBeginTag+1,
indexOfCloseTag2);

               outXML.append(">"+"</"+tagName);

               newIndexOfBeginTag = inXML.indexOf("<", indexOfCloseTag2);
               outXML.append(inXML.substring(indexOfCloseTag2+1,
newIndexOfBeginTag));
               indexOfBeginTag = newIndexOfBeginTag;
            }
            else { // we have a tag closing with ">"
```

```
                // copy the original inXML to outXML till the indexOfCloseTag1
                outXML.append(inXML.substring(indexOfBeginTag, indexOfCloseTag1));

                outXML.append(">");

                newIndexOfBeginTag = inXML.indexOf("<", indexOfCloseTag1);

                if(newIndexOfBeginTag != -1)
                   outXML.append(inXML.substring(indexOfCloseTag1+1,
newIndexOfBeginTag));
                else
                   outXML.append(inXML.substring(indexOfCloseTag1+1));
                indexOfBeginTag = newIndexOfBeginTag;
             }

             if(indexOfBeginTag < 0)
                break;
          }

          return outXML.toString();
       }
       catch(Exception e) {
          e.printStackTrace();
          return inXML;
       }
    }


    /**
     * Test stub
     *
     */
    public static void main(String argv[]) {
       try {
          FileInputStream fis = new FileInputStream("test.xml");
          int fileSize = fis.available();
          byte[] fileBytes= new byte[fileSize];
          fis.read(fileBytes);
          fis.close();
          String fileString = new String(fileBytes);
          System.out.println("Before="+fileString+"*********");
          System.out.println("After="+modifyXMLTags(fileString)+"*********");
       }
       catch(Exception e) {
          e.printStackTrace();
       }
    }
}
```

```
/* -*- Mode: java -*-
 *
 * Copyright (c) 1996 International Business Machines, Corp.
 * All rights reserved.
 *
 * IBM disclaims all warranties with regard to this software,
 * including all implied warranties of merchantability and fitness,
 * in no event shall IBM be liable for any special, indirect or
 * consequential damages or any damages whatsoever resulting from loss of
 * use, data or profits, whether in an action of contract, negligence or
 * other tortious action, arising out of or in connection with the use
 * or performance of this software.
 *
 * Comments and additions should be sent to:
 *
 *      Scott Spangler
 *      IBM Almaden Research Center
 *      650 Harry Road
 *      San Jose, CA 95123
 *      (408) 927-2887
 *
 * $Author: spangles $
 * $Source: /user/dbmine/core/cvs/master/cv/StringVector.java,v $
 * $Revision: 1.6 $
 * $Date:            19:19:18 $
 * $State: Exp $
 *
 */


package com.ibm.xmlgateway;

import java.util.*;
import java.io.*;

public class StringVector extends Vector {

public StringVector() {

        super();

}


public StringVector(String s) {
  super();
   StringTokenizer st = new StringTokenizer(s,",");

  while (st.hasMoreTokens()) {
    String ss =st.nextToken();
    addElement(ss.trim());
  }
}

public StringVector(String s,String tok) {
  super();
   StringTokenizer st = new StringTokenizer(s,tok);

  while (st.hasMoreTokens()) {
    String ss =st.nextToken();
    addElement(ss.trim());
  }
}
```

```
public StringVector(StringVector sv) {
  super();
  for (int i=0; i<sv.size(); i++) {
    addElement(sv.myElementAt(i));
  }
}

public String myElementAt(int i) {

        String temp = (String)super.elementAt(i);
        return(temp);

}



public boolean contains(String x) {
        for (int j=0; j< size(); j++) {
                if (myElementAt(j).equals(x)) {
                        return(true);
                }
        }
        return(false);
}

public void remove(String x) {
  for (int j=0; j< size(); j++) {
    if (myElementAt(j).equals(x)) {
      removeElementAt(j);
      return;
    }
  }
}

public int positionOf(String x) {
  for (int j=0; j<size(); j++) {
    if (myElementAt(j).equalsIgnoreCase(x)) return(j);
  }
  return(-1);
}

public String[] getStringArray() {
  String result[] = new String[size()];
  for (int i=0; i<(size()); i++) {
    result[i] = myElementAt(i);
  }
  return(result);
}

public String makeString() {
  String result = "";
  for (int i=0; i<(size()-1); i++) {
    result = result + myElementAt(i) + ",";
  }
  result = result + myElementAt(size()-1);
  return(result);
}

public void removeSubstringsOf(String s) {
  for (int j=0; j< size(); j++) {
    if (j>=size()) return;
    if (s.indexOf(myElementAt(j))>-1) {
```

```
      //     System.out.println("removing " + myElementAt(j) + " since it conflicts
with " + s);
      removeElementAt(j);
    }
  }
}

public boolean containsSubstring(String s) {
  for (int j=0; j<size(); j++) {
    if (myElementAt(j).indexOf(s)>-1) {
      return(true);
    }
  }
  return(false);
}

public StringVector removeDuplicates() {
  StringVector result = new StringVector();
  for (int i=0; i<size(); i++) {
    String s = myElementAt(i);
    if (!result.contains(s)) result.addElement(s);
  }
  return(result);
}

public static void main(String args[]) {
  String s = "sepal length, sepal width,petal length,petal width,class";
  StringVector sv = new StringVector(s);
  for (int i=0; i<sv.size(); i++)
    System.out.println(sv.myElementAt(i));
}


}
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

// import java classes
import java.io.*;
import java.net.*;
import java.util.*;

// import IBM XML for Java parser classes
import com.ibm.xml.parser.*;

// import W3C classes
import org.w3c.dom.*;

/**
 * This utility class walks a TXDocument(specified using the setTXDocument() method)
 * and returns a Vector of String's of the values for a certain tag name.
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

public class TagValueExtractor {
   private static TXDocument txd;

   private static boolean DEBUG = false;  // used for debugging when running with
the main() method

   /**
    * @param xmlString the XML document in a String
    */
   public static void setTXDocument(String xmlString) {

      try {
         Parser p = new Parser("da-da-da");  // do I need a valid name in there?
this is not!
         p.setProcessNamespace(true);

         StringReader sr = new StringReader(xmlString);
         txd = p.readStream(sr);
      }
      catch(Exception e) {
         System.out.println("problem in setting txdocument: "+e.toString());
         e.printStackTrace();
      }
   }

   /**
    * @param _txd the TXDocument to walk for the tag names.
    */
   public static void setTXDocument(TXDocument _txd) {
      txd = _txd;
   }

   /**
    * @return a vector containing String's(trimmed) for every value of the tag name
found in the
    * XML document. If that tag is not found, then the Vector returned is of size
zero,
```

```
     * never null.
     *
     * @param tagName the name of the tag for which to look for values in the last
TXDocument
     * specified using the setTXDocument() method.
     *
     */
    public static Vector getTagValues(String tagName) {
       NodeList nl = txd.getElementsByTagName(tagName);
       Node node;
       Vector valuesVec = new Vector();

       for(int i=0; i<nl.getLength(); i++) {
          node = nl.item(i);
          valuesVec.addElement(((TXElement)node).getText().trim());
       }

       return valuesVec;
    }

    /**
     * @return a String(trimmed) containing first value of the tag name found in the
XML document.
     * If that tag is not found, then returns null.
     *
     * @param tagName the name of the tag for which to look for a value in the last
TXDocument
     * specified using the setTXDocument() method.
     *
     */
    public static String getTagFirstValue(String tagName) {
       NodeList nl = txd.getElementsByTagName(tagName);
       Node node;

       if(nl.getLength() > 0) {
          node = nl.item(0);

          if(DEBUG)
          System.out.println(((TXElement)node).getText().trim());

          return((TXElement)node).getText().trim();
       }
       else
          return null;
    }

    public static void main(String argv[]) {
       String xml = "<?xml version=\"1.0\" encoding=\"UTF-8\"?> <TSD_INBOUND_TICKET>
<GATEWAY> <GATEWAY.NEW> 1 </GATEWAY.NEW> <GATEWAY.NAME> TSD4.2GatewayA
</GATEWAY.NAME> <GATEWAY.BRIDGE> LINDYP </GATEWAY.BRIDGE> <GATEWAY.TIMESTAMP> Wed
May 03 16:33:41 PDT 2000 </GATEWAY.TIMESTAMP> </GATEWAY> <PROBLEM_CLOSURE>
<PROBLEM_CLOSURE.CLOSURE_ID> _ </PROBLEM_CLOSURE.CLOSURE_ID>
<PROBLEM_CLOSURE.TRANSACTIONTYPE> 1 </PROBLEM_CLOSURE.TRANSACTIONTYPE>
<PROBLEM_CLOSURE.CREATECALL> 1 </PROBLEM_CLOSURE.CREATECALL>
<PROBLEM_CLOSURE.CREATEPROBLEM> 0 </PROBLEM_CLOSURE.CREATEPROBLEM>
<PROBLEM_CLOSURE.CLOSEPROBLEM> 1 </PROBLEM_CLOSURE.CLOSEPROBLEM>
<PROBLEM_CLOSURE.PICKUPDISPATCH> 0 </PROBLEM_CLOSURE.PICKUPDISPATCH>
<PROBLEM_CLOSURE.TRANSFERUSER> 0 </PROBLEM_CLOSURE.TRANSFERUSER>
<PROBLEM_CLOSURE.DONOTIFICATION> 0 </PROBLEM_CLOSURE.DONOTIFICATION>
<PROBLEM_CLOSURE.EXPLODEGROUP> 0 </PROBLEM_CLOSURE.EXPLODEGROUP>
<PROBLEM_CLOSURE.SOLUTIONMETHOD> _ </PROBLEM_CLOSURE.SOLUTIONMETHOD>
<PROBLEM_CLOSURE.PROBLEM_ID> _ </PROBLEM_CLOSURE.PROBLEM_ID>
<PROBLEM_CLOSURE.CALL_ID> _ </PROBLEM_CLOSURE.CALL_ID> <PROBLEM_CLOSURE.SESSION_ID>
```

```
_ </PROBLEM_CLOSURE.SESSION_ID> <PROBLEM_CLOSURE.DISPATCH_ID> _
</PROBLEM_CLOSURE.DISPATCH_ID> <PROBLEM_CLOSURE.MODIFY_DATETIME> 1
</PROBLEM_CLOSURE.MODIFY_DATETIME> <PROBLEM_CLOSURE.SESSION_BEGIN_DATE> _
</PROBLEM_CLOSURE.SESSION_BEGIN_DATE> <PROBLEM_CLOSURE.SESSION_BEGIN_TIME> _
</PROBLEM_CLOSURE.SESSION_BEGIN_TIME> <PROBLEM_CLOSURE.SESSION_END_DATE> _
</PROBLEM_CLOSURE.SESSION_END_DATE> <PROBLEM_CLOSURE.SESSION_END_TIME> _
</PROBLEM_CLOSURE.SESSION_END_TIME> <PROBLEM_CLOSURE.LOCATION_ID> _
</PROBLEM_CLOSURE.LOCATION_ID> <PROBLEM_CLOSURE.CALLER_ID> _
</PROBLEM_CLOSURE.CALLER_ID> <PROBLEM_CLOSURE.CALLER_NAME> fannon
</PROBLEM_CLOSURE.CALLER_NAME> <PROBLEM_CLOSURE.CALLER_PHONE> _
</PROBLEM_CLOSURE.CALLER_PHONE> <PROBLEM_CLOSURE.USER_ID> LINDYPUSER
</PROBLEM_CLOSURE.USER_ID> <PROBLEM_CLOSURE.CALL_CODE> LINDYPINCOMING
</PROBLEM_CLOSURE.CALL_CODE> <PROBLEM_CLOSURE.SEVERITY> 4
</PROBLEM_CLOSURE.SEVERITY> <PROBLEM_CLOSURE.PROBLEM_CODE> TRANSFERRED
</PROBLEM_CLOSURE.PROBLEM_CODE> <PROBLEM_CLOSURE.PROBLEM_TYPE> LINDYP TSD4.2GatewayA
</PROBLEM_CLOSURE.PROBLEM_TYPE> <PROBLEM_CLOSURE.SYSTEM> HARDWARE
</PROBLEM_CLOSURE.SYSTEM> <PROBLEM_CLOSURE.COMPONENT> Personal System
</PROBLEM_CLOSURE.COMPONENT> <PROBLEM_CLOSURE.ITEM> processor
</PROBLEM_CLOSURE.ITEM> <PROBLEM_CLOSURE.MODULE> intel </PROBLEM_CLOSURE.MODULE>
<PROBLEM_CLOSURE.DESCRIPTION> Decriptionnnnn </PROBLEM_CLOSURE.DESCRIPTION>
<PROBLEM_CLOSURE.SERIAL_NUMBER> _ </PROBLEM_CLOSURE.SERIAL_NUMBER>
<PROBLEM_CLOSURE.INVENTORY_ID> _ </PROBLEM_CLOSURE.INVENTORY_ID>
<PROBLEM_CLOSURE.PROBLEM_RESULT> _ </PROBLEM_CLOSURE.PROBLEM_RESULT>
<PROBLEM_CLOSURE.TIME_SPENT> _ </PROBLEM_CLOSURE.TIME_SPENT>
<PROBLEM_CLOSURE.DIAG_NODE> _ </PROBLEM_CLOSURE.DIAG_NODE>
<PROBLEM_CLOSURE.FLX_CAL_VCHR1> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR1>
<PROBLEM_CLOSURE.FLX_CAL_VCHR2> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR2>
<PROBLEM_CLOSURE.FLX_CAL_VCHR3> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR3>
<PROBLEM_CLOSURE.FLX_CAL_VCHR4> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR4>
<PROBLEM_CLOSURE.FLX_CAL_INT1> _ </PROBLEM_CLOSURE.FLX_CAL_INT1>
<PROBLEM_CLOSURE.FLX_CAL_INT2> _ </PROBLEM_CLOSURE.FLX_CAL_INT2>
<PROBLEM_CLOSURE.FLX_CAL_INT3> _ </PROBLEM_CLOSURE.FLX_CAL_INT3>
<PROBLEM_CLOSURE.FLX_CAL_INT4> _ </PROBLEM_CLOSURE.FLX_CAL_INT4>
<PROBLEM_CLOSURE.FLX_CAL_DATE1> _ </PROBLEM_CLOSURE.FLX_CAL_DATE1>
<PROBLEM_CLOSURE.FLX_CAL_DATE2> _ </PROBLEM_CLOSURE.FLX_CAL_DATE2>
<PROBLEM_CLOSURE.FLX_CAL_TIME1> _ </PROBLEM_CLOSURE.FLX_CAL_TIME1>
<PROBLEM_CLOSURE.FLX_CAL_TIME2> _ </PROBLEM_CLOSURE.FLX_CAL_TIME2>
<PROBLEM_CLOSURE.FLX_PRO_VCHR1> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR1>
<PROBLEM_CLOSURE.FLX_PRO_VCHR2> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR2>
<PROBLEM_CLOSURE.FLX_PRO_VCHR3> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR3>
<PROBLEM_CLOSURE.FLX_PRO_VCHR4> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR4>
<PROBLEM_CLOSURE.FLX_PRO_INT1> _ </PROBLEM_CLOSURE.FLX_PRO_INT1>
<PROBLEM_CLOSURE.FLX_PRO_INT2> _ </PROBLEM_CLOSURE.FLX_PRO_INT2>
<PROBLEM_CLOSURE.FLX_PRO_INT3> _ </PROBLEM_CLOSURE.FLX_PRO_INT3>
<PROBLEM_CLOSURE.FLX_PRO_INT4> _ </PROBLEM_CLOSURE.FLX_PRO_INT4>
<PROBLEM_CLOSURE.FLX_PRO_DATE1> _ </PROBLEM_CLOSURE.FLX_PRO_DATE1>
<PROBLEM_CLOSURE.FLX_PRO_DATE2> _ </PROBLEM_CLOSURE.FLX_PRO_DATE2>
<PROBLEM_CLOSURE.FLX_PRO_TIME1> _ </PROBLEM_CLOSURE.FLX_PRO_TIME1>
<PROBLEM_CLOSURE.FLX_PRO_TIME2> _ </PROBLEM_CLOSURE.FLX_PRO_TIME2>
<PROBLEM_CLOSURE.RCV_GROUP_ID> _ </PROBLEM_CLOSURE.RCV_GROUP_ID>
<PROBLEM_CLOSURE.RCV_USER_ID> _ </PROBLEM_CLOSURE.RCV_USER_ID>
<PROBLEM_CLOSURE.LINE_NUMBER> _ </PROBLEM_CLOSURE.LINE_NUMBER>
<PROBLEM_CLOSURE.NOTIFICATION_DATE> _ </PROBLEM_CLOSURE.NOTIFICATION_DATE>
<PROBLEM_CLOSURE.NOTIFICATION_TIME> _ </PROBLEM_CLOSURE.NOTIFICATION_TIME>
<PROBLEM_CLOSURE.NOTIFICATION_TYPE> _ </PROBLEM_CLOSURE.NOTIFICATION_TYPE>
<PROBLEM_CLOSURE.NOTIFY_CONTACT> _ </PROBLEM_CLOSURE.NOTIFY_CONTACT>
<PROBLEM_CLOSURE.SOLUTION> _ </PROBLEM_CLOSURE.SOLUTION> <PROBLEM_CLOSURE.ACTIVE> 0
</PROBLEM_CLOSURE.ACTIVE> <PROBLEM_CLOSURE.SOLUTION_ID> _
</PROBLEM_CLOSURE.SOLUTION_ID> <PROBLEM_CLOSURE.AID_TYPE> _
</PROBLEM_CLOSURE.AID_TYPE> <PROBLEM_CLOSURE.ANNOTATION_FILE> _
</PROBLEM_CLOSURE.ANNOTATION_FILE> <PROBLEM_CLOSURE.CONTROL_TIME> _
</PROBLEM_CLOSURE.CONTROL_TIME> <PROBLEM_CLOSURE.FLX_PRO_VCHR5> _
</PROBLEM_CLOSURE.FLX_PRO_VCHR5> <PROBLEM_CLOSURE.GROUP_ID> LINDYP
```

```
</PROBLEM_CLOSURE.GROUP_ID> <PROBLEM_CLOSURE.LOGGED_USER> LINDYP USER
</PROBLEM_CLOSURE.LOGGED_USER> </PROBLEM_CLOSURE> <PROBLEM_CLOSURE2>
<PROBLEM_CLOSURE2.CLOSURE_ID> _ </PROBLEM_CLOSURE2.CLOSURE_ID>
<PROBLEM_CLOSURE2.DOCUMENT_ID> _ </PROBLEM_CLOSURE2.DOCUMENT_ID>
<PROBLEM_CLOSURE2.VEND_FIELD1> _ </PROBLEM_CLOSURE2.VEND_FIELD1>
<PROBLEM_CLOSURE2.VEND_FIELD2> _ </PROBLEM_CLOSURE2.VEND_FIELD2>
<PROBLEM_CLOSURE2.VEND_FIELD3> _ </PROBLEM_CLOSURE2.VEND_FIELD3>
<PROBLEM_CLOSURE2.VEND_FIELD4> _ </PROBLEM_CLOSURE2.VEND_FIELD4>
<PROBLEM_CLOSURE2.VEND_FIELD5> _ </PROBLEM_CLOSURE2.VEND_FIELD5>
<PROBLEM_CLOSURE2.VEND_FIELD6> _ </PROBLEM_CLOSURE2.VEND_FIELD6>
<PROBLEM_CLOSURE2.VEND_FIELD7> _ </PROBLEM_CLOSURE2.VEND_FIELD7>
<PROBLEM_CLOSURE2.VEND_FIELD8> _ </PROBLEM_CLOSURE2.VEND_FIELD8>
<PROBLEM_CLOSURE2.VEND_FIELD9> _ </PROBLEM_CLOSURE2.VEND_FIELD9>
<PROBLEM_CLOSURE2.VEND_FIELD10> _ </PROBLEM_CLOSURE2.VEND_FIELD10>
<PROBLEM_CLOSURE2.DATETIME1> _ </PROBLEM_CLOSURE2.DATETIME1>
<PROBLEM_CLOSURE2.DATETIME2> _ </PROBLEM_CLOSURE2.DATETIME2>
<PROBLEM_CLOSURE2.DATETIME3> _ </PROBLEM_CLOSURE2.DATETIME3>
<PROBLEM_CLOSURE2.DATETIME4> _ </PROBLEM_CLOSURE2.DATETIME4>
<PROBLEM_CLOSURE2.DATETIME5> _ </PROBLEM_CLOSURE2.DATETIME5>
<PROBLEM_CLOSURE2.DATETIME6> _ </PROBLEM_CLOSURE2.DATETIME6>
<PROBLEM_CLOSURE2.DATETIME7> _ </PROBLEM_CLOSURE2.DATETIME7>
<PROBLEM_CLOSURE2.DATETIME8> _ </PROBLEM_CLOSURE2.DATETIME8>
<PROBLEM_CLOSURE2.DATETIME9> _ </PROBLEM_CLOSURE2.DATETIME9>
<PROBLEM_CLOSURE2.DATETIME10> _ </PROBLEM_CLOSURE2.DATETIME10>
<PROBLEM_CLOSURE2.BRIDGE_ID> LINDYP </PROBLEM_CLOSURE2.BRIDGE_ID>
<PROBLEM_CLOSURE2.BRIDGE_TICKET_NO> 00002864 </PROBLEM_CLOSURE2.BRIDGE_TICKET_NO>
<PROBLEM_CLOSURE2.INCOMING_FLAG> 0 </PROBLEM_CLOSURE2.INCOMING_FLAG>
<PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION> LINDYP </PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION>
<PROBLEM_CLOSURE2.CUSTOMER_ID> _ </PROBLEM_CLOSURE2.CUSTOMER_ID>
<PROBLEM_CLOSURE2.ACCOUNT_ID> _ </PROBLEM_CLOSURE2.ACCOUNT_ID>
<PROBLEM_CLOSURE2.REPORTER_ID> _ </PROBLEM_CLOSURE2.REPORTER_ID>
<PROBLEM_CLOSURE2.RESOLVER_ID> _ </PROBLEM_CLOSURE2.RESOLVER_ID>
<PROBLEM_CLOSURE2.REPORTER_GROUP> _ </PROBLEM_CLOSURE2.REPORTER_GROUP>
<PROBLEM_CLOSURE2.RESOLVER_GROUP> _ </PROBLEM_CLOSURE2.RESOLVER_GROUP>
<PROBLEM_CLOSURE2.TEAM> _ </PROBLEM_CLOSURE2.TEAM> <PROBLEM_CLOSURE2.CALLBACK_DATE>
_ </PROBLEM_CLOSURE2.CALLBACK_DATE> <PROBLEM_CLOSURE2.CALLBACK_TIME> _
</PROBLEM_CLOSURE2.CALLBACK_TIME> <PROBLEM_CLOSURE2.ORIG_TARGET_DATE> _
</PROBLEM_CLOSURE2.ORIG_TARGET_DATE> <PROBLEM_CLOSURE2.ORIG_TARGET_TIME> _
</PROBLEM_CLOSURE2.ORIG_TARGET_TIME> <PROBLEM_CLOSURE2.CURR_TARGET_DATE> _
</PROBLEM_CLOSURE2.CURR_TARGET_DATE> <PROBLEM_CLOSURE2.CURR_TARGET_TIME> _
</PROBLEM_CLOSURE2.CURR_TARGET_TIME> <PROBLEM_CLOSURE2.OCCURRED_DATE> 2000-05-03
</PROBLEM_CLOSURE2.OCCURRED_DATE> <PROBLEM_CLOSURE2.OCCURRED_TIME> 23:33:52
</PROBLEM_CLOSURE2.OCCURRED_TIME> <PROBLEM_CLOSURE2.SOLVED_DATE> _
</PROBLEM_CLOSURE2.SOLVED_DATE> <PROBLEM_CLOSURE2.SOLVED_TIME> _
</PROBLEM_CLOSURE2.SOLVED_TIME> <PROBLEM_CLOSURE2.CAUSE_CHANGE_NO> _
</PROBLEM_CLOSURE2.CAUSE_CHANGE_NO> <PROBLEM_CLOSURE2.ORIGINAL_SEVERITY> 4
</PROBLEM_CLOSURE2.ORIGINAL_SEVERITY> <PROBLEM_CLOSURE2.DURATION> _
</PROBLEM_CLOSURE2.DURATION> <PROBLEM_CLOSURE2.NODEID> _ </PROBLEM_CLOSURE2.NODEID>
<PROBLEM_CLOSURE2.REASSIGNMENT> _ </PROBLEM_CLOSURE2.REASSIGNMENT>
<PROBLEM_CLOSURE2.PROBLEM_ABSTRACT> Decriptionnnnn
</PROBLEM_CLOSURE2.PROBLEM_ABSTRACT> <PROBLEM_CLOSURE2.PROBLEM_DUPLICATE> _
</PROBLEM_CLOSURE2.PROBLEM_DUPLICATE> <PROBLEM_CLOSURE2.RCA_REQUIRED> _
</PROBLEM_CLOSURE2.RCA_REQUIRED> <PROBLEM_CLOSURE2.PREV_OCCURRED_DATE> _
</PROBLEM_CLOSURE2.PREV_OCCURRED_DATE> <PROBLEM_CLOSURE2.PREV_OCCURRED_TIME> _
</PROBLEM_CLOSURE2.PREV_OCCURRED_TIME> <PROBLEM_CLOSURE2.PREV_CT_DATE> _
</PROBLEM_CLOSURE2.PREV_CT_DATE> <PROBLEM_CLOSURE2.PREV_CT_TIME> _
</PROBLEM_CLOSURE2.PREV_CT_TIME> <PROBLEM_CLOSURE2.CAUSE_CODE> _
</PROBLEM_CLOSURE2.CAUSE_CODE> <PROBLEM_CLOSURE2.RCA_TEXT> _
</PROBLEM_CLOSURE2.RCA_TEXT> <PROBLEM_CLOSURE2.REASON> _ </PROBLEM_CLOSURE2.REASON>
</PROBLEM_CLOSURE2> <HISTORY> <WORK_HISTORY.WORK_ID> _ </WORK_HISTORY.WORK_ID>
<WORK_HISTORY.PROBLEM_ID> _ </WORK_HISTORY.PROBLEM_ID> <WORK_HISTORY.USER_ID>
LINDYPUSER </WORK_HISTORY.USER_ID> <WORK_HISTORY.WORK_BEGIN_DATE> 2000-05-03
</WORK_HISTORY.WORK_BEGIN_DATE> <WORK_HISTORY.WORK_BEGIN_TIME> 23:34:39
```

```
</WORK_HISTORY.WORK_BEGIN_TIME> <WORK_HISTORY.WORK_END_DATE> 2000-05-03
</WORK_HISTORY.WORK_END_DATE> <WORK_HISTORY.WORK_END_TIME> 23:34:39
</WORK_HISTORY.WORK_END_TIME> <WORK_HISTORY.DESCRIPTION> hello
</WORK_HISTORY.DESCRIPTION> <WORK_HISTORY.DESC_OVRFLW> _ </WORK_HISTORY.DESC_OVRFLW>
<WORK_HISTORY.MODIFY_DATETIME> 1 </WORK_HISTORY.MODIFY_DATETIME>
<WORK_HISTORY.TIME_STAMP> 2000-05-03 19:34:52.351299 </WORK_HISTORY.TIME_STAMP>
<WORK_HISTORY.ACTIVITY_ACTION_ID> _ </WORK_HISTORY.ACTIVITY_ACTION_ID> </HISTORY>
<HISTORY> <WORK_HISTORY.WORK_ID> _ </WORK_HISTORY.WORK_ID> <WORK_HISTORY.PROBLEM_ID>
_ </WORK_HISTORY.PROBLEM_ID> <WORK_HISTORY.USER_ID> LINDYPUSER
</WORK_HISTORY.USER_ID> <WORK_HISTORY.WORK_BEGIN_DATE> 2000-05-03
</WORK_HISTORY.WORK_BEGIN_DATE> <WORK_HISTORY.WORK_BEGIN_TIME> 23:34:54
</WORK_HISTORY.WORK_BEGIN_TIME> <WORK_HISTORY.WORK_END_DATE> 2000-05-03
</WORK_HISTORY.WORK_END_DATE> <WORK_HISTORY.WORK_END_TIME> 23:34:54
</WORK_HISTORY.WORK_END_TIME> <WORK_HISTORY.DESCRIPTION> there
</WORK_HISTORY.DESCRIPTION> <WORK_HISTORY.DESC_OVRFLW> _ </WORK_HISTORY.DESC_OVRFLW>
<WORK_HISTORY.MODIFY_DATETIME> 1 </WORK_HISTORY.MODIFY_DATETIME>
<WORK_HISTORY.TIME_STAMP> 2000-05-03 19:35:02.504746 </WORK_HISTORY.TIME_STAMP>
<WORK_HISTORY.ACTIVITY_ACTION_ID> _ </WORK_HISTORY.ACTIVITY_ACTION_ID> </HISTORY>
<HISTORY> <WORK_HISTORY.WORK_ID> _ </WORK_HISTORY.WORK_ID> <WORK_HISTORY.PROBLEM_ID>
_ </WORK_HISTORY.PROBLEM_ID> <WORK_HISTORY.USER_ID> LINDYPUSER
</WORK_HISTORY.USER_ID> <WORK_HISTORY.WORK_BEGIN_DATE> _
</WORK_HISTORY.WORK_BEGIN_DATE> <WORK_HISTORY.WORK_BEGIN_TIME> _
</WORK_HISTORY.WORK_BEGIN_TIME> <WORK_HISTORY.WORK_END_DATE> _
</WORK_HISTORY.WORK_END_DATE> <WORK_HISTORY.WORK_END_TIME> _
</WORK_HISTORY.WORK_END_TIME> <WORK_HISTORY.DESCRIPTION> TSD4.2GatewayA:
ACTIVE_SLA_.ACTIVE_SLA_ID = _: ACTIVE_SLA.TERM_ID = _: ACTIVE_SLA.REFERENCE_ID = _:
ACTIVE_SLA.RELATIVE_TO = _: ACTIVE_SLA.SLA_STATE = _: ACTIVE_SLA.BREACH_DATE = _:
ACTIVE_SLA.BREACH_TIME = _: ACTIVE_SLA.NEXT_SCHEDULE_ID = _:
ACTIVE_SLA.NEXT_FIRE_DATE = _: ACTIVE_SLA.NEXT_FIRE_TIME = _
</WORK_HISTORY.DESCRIPTION> <WORK_HISTORY.DESC_OVRFLW> _ </WORK_HISTORY.DESC_OVRFLW>
<WORK_HISTORY.MODIFY_DATETIME> 1 </WORK_HISTORY.MODIFY_DATETIME>
<WORK_HISTORY.TIME_STAMP> _ </WORK_HISTORY.TIME_STAMP>
<WORK_HISTORY.ACTIVITY_ACTION_ID> _ </WORK_HISTORY.ACTIVITY_ACTION_ID> </HISTORY>
</TSD_INBOUND_TICKET>";
        setTXDocument(xml);
        DEBUG = true;
        getTagFirstValue(argv[0]);
    }
}
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

// import java classes
import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;

// import java servlet classes
import javax.servlet.*;
import javax.servlet.http.*;

// import xmlbridge classes

// import IBM XML for Java parser classes
import com.ibm.xml.parser.*;

// import W3C classes
import org.w3c.dom.*;

// import classes from Bridge for common use
import com.ibm.xmlbridge.*;

/**
 * This class serves as the base class for new Gateways that
 * recieve data from the bridge and pass it onto the destination
 * database and vice-versa through the Bridge. It is implemented
 * as a servlet that runs under a servlet runner like IBM WebSphere2.0
 * and connects on one side to the Bridge using a generic protocol and
 * on the other side to a backend database system like TSD/DB2.
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

abstract public class XMLGWforANYServlet extends HttpServlet {

   /**
    * the name/id of this Gateway
    */
   public String gatewayName = "NOGatewayNameYet";                // the initial
name and default if one does not exist in the props file

   /**
    * the name/id of remote Gateway
    */
   public String remoteGatewayName = "RemoteGatewayName_Default"; // the initial
name and default if one does not exist in the props file

   /**
    * the DB connector object that allows the Gateway to execute queries.
    */
   public DBConnector dbc;

   /**
    * the logger object used to send output/error msg's
    */
   public XMLGatewayLogger logger;
```

```
   /**
    * the bridge communicator object used to send records to the Bridge
    */
   public XMLBridgeCommunicator xmlbc;

   /**
    *
    * The servlet's initialization method. It gets inherited by the deriving class
and should NOT be
    * overridden but just left as it it.
    *
    */
   public void init(ServletConfig config) throws ServletException {
      super.init(config);
      String xmlgatewayDirString = config.getInitParameter("xmlgatewaydir");
      startUpGeneric(xmlgatewayDirString);
   }

   /**
    * Does a bunch of startup one time stuff. It is called by the init() method
internally. The developer
    * must override the startUpSpecific() method to do additional startup stuff like
loading parameters from
    * a properties file, etc.
    *
    * @param xmlgatewayDirString the base directory for the XMLGateway files without
the ending '\' or '/'.
    * this string is picked up from the servlet's initializations parameters by the
init() method and passed
    * on to the this method automatically.
    *
    */
   private void startUpGeneric(String xmlgatewayDirString) throws ServletException {
      File xmlgatewayDir;
      // create the logger for logging purposes
      logger = new XMLGatewayLogger();

      // get the directory containing the XMLGateway stuff
      if(xmlgatewayDirString != null) {
         xmlgatewayDir = new File(xmlgatewayDirString);
      }
      else {
         throw new ServletException("xmlgatewayDirString, the init parameter, is
null");
      }

      if(xmlgatewayDir!=null) {
         if(xmlgatewayDir.exists()) {
            if(!(xmlgatewayDir.isDirectory())) {
               throw new ServletException("not a
directory:"+xmlgatewayDir.getAbsolutePath());
            }
         }
      }

      // call the method to to initialization stuff for the specific derived class
      startUpSpecific(xmlgatewayDirString);
   }

   /**
    * The developer must override the startUpSpecific() method to do additional
startup stuff like loading parameters
```

```
     * from a properties file, etc.
     *
     * @param xmlgatewayDirString the base directory for the XMLGateway files without
the ending '\' or '/'.
     * this string is picked up from the servlet's initializations parameters by the
init() method and passed
     * on to the this method automatically.
     *
     */
    abstract public void startUpSpecific(String xmlgatewayDirString);

    /**
     * Escapes the single quote in an SQL string by replacing it with two single
quotes so that the SQL
     * becomes acceptable.
     *
     * @param value the string(SQL) containing the single quote.
     *
     * @return value the string(SQL) containing the escaped single quote.
     */
    public String escapeSingleQuote(String value) {
        if(value == null)
            return value;

        int indexOfQuote = value.indexOf("'");
        if(indexOfQuote == -1)
            return value;
        else {
            StringBuffer sb = new StringBuffer();
            for(int i=0; i < value.length(); i++) {
                if(value.charAt(i) == '\'') {
                    sb.append('\'');
                    sb.append(value.charAt(i));
                }
                else {
                    sb.append(value.charAt(i));
                }
            }
            return sb.toString();
        }
    }

    /**
     * This method is left as an abstract method and needs to be implemented in the
deriving class
     * specifically for the target database. The inbound data(a problem ticket for
example) is sent
     * in XML to this method which is then parsed to get the data for CREATING a new
record(a new
     * problem ticket for example) into the database.
     *
     * @param createRecordXML the string(SQL) containing the single quote.
     * @param remoteId the record Id in the remote database.
     * @param localId the record Id in the local database.
     *
     * @return a boolean indicating whether or not the request was successfully
completed.
     */
    abstract public boolean createRecordInDB(String createRecordXML, String remoteId,
String localId);

    /**
     * This method is left as an abstract method and needs to be implemented in the
```

```
deriving class
     * specifically for the target database. The inbound data(a problem ticket for
example) is sent
     * in XML to this method which is then parsed to get the data for UPDATING a new
record(a new
     * problem ticket for example) into the database.
     *
     * @param updateRecordXML the string(SQL) containing the single quote.
     * @param remoteId the record Id in the remote database.
     * @param localId the record Id in the local database.
     *
     * @return a boolean indicating whether or not the request was successfully
completed.
     */
    abstract public boolean updateRecordInDB(String updateRecordXML, String remoteId,
String localId);

    /**
     * This method is left as an abstract method and needs to be implemented in the
deriving class
     * specifically for the target database. It can be used to associate the
record(ticket) number in
     * this Gateway's database with the record(ticket) number of the ticket in the
remote database by
     * updating the record(ticket) in this Gateway's database with the
remoteRecordNumber.
     *
     * @param remoteId the record Id in the remote database.
     * @param localId the record Id in the local database.
     *
     * @return a boolean indicating whether or not the request was successfully
completed.
     */
    abstract public boolean ackRecordInDB(String remoteId, String localId);

    /**
     * This method sends new a record to the Bridge that is newly created in the
local database.
     * The Bridge will then assign it a remote record number from the record stock of
remote database.
     *
     * @param recordId the record id in the local database.
     * @param outboundXML the XML document in a String contaning the data for the
record.
     *
     */
    public boolean sendNewRecordToBridge(String recordId, String outboundXML) {
        try {
            return xmlbc.sendTicket(recordId, gatewayName, outboundXML);
        }
        catch(Exception e) {
            log("problem in sending record to the Bridge:"+e.toString());
            e.printStackTrace();
            return false;
        }
    }

    /**
     * This method sends an updated record to the Bridge that has just been updated
in the local database.
     *
     * @param recordId the record id in the attached source database.
     * @param outboundXML the XML document in a String contaning the data for the
```

```
record.
     *
     */
    public boolean sendUpdatedRecordToBridge(String recordId, String
remoteTicketNumber, String outboundXML) {
       try {
          return xmlbc.updateTicket(recordId, gatewayName, remoteTicketNumber,
outboundXML);
       }
       catch(Exception e) {
          log("problem in sending record to the Bridge:"+e.toString());
          e.printStackTrace();
          return false;
       }
    }

    /**
     * this method takes a string, containing for example a problem id, and
     * left pads it with zeros to make it 8 characters in total length. If
     * the initial string is 8 characters or more in length, then it is returned
     * as it is.
     *
     * @param localId the initial string.
     *
     * @return a the 8 digit, left padded with zeros, problemId
     */
    public String modifyProblemId(String localId) {
       // ensure localId is 8 digits left padded with 0's
       int incomingSize = localId.length();
       if(incomingSize < 8) {
          int requiredZeros = 8-incomingSize;
          for(int i=0; i<requiredZeros; i++) {
             localId = "0"+localId;
          }
       }

       return localId;
    }

    /**
     * This method can be overriden to send output and error messages to a desired
stream. By deafult it
     * prints it out to the standard output.
     *
     * @param msg the output or error message for logging purposes.
     *
     */
    public void log(String msg) {
       if(logger != null)
          logger.log(gatewayName, Syslog.DEBUG, msg);
       else  // use stdout by default
          System.out.println(gatewayName+":"+Syslog.DEBUG+":"+msg);

    }

    /**
     * this method can be overridden to do any house cleaning stuff like closing
sockets,
     * database connections, etc.
     *
     * @param msg the output or error message for logging purposes.
     *
     */
```

```
    public void destroy() {
       super.destroy();
    }
}
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

// import java classes
import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.*;
import java.lang.*;

/**
 * This class serves as the base class for classes that provide
 * the functionality to connect to specific databases using JDBC
 * and interacting with them for various purposes.
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

abstract public class DBConnector extends Thread {

   /**
    * the servlet object that instantiates an object of this class, i.e., the
    * main Gateway servlet that has an object of this class.
    */
   public XMLGWforANYServlet servlet;


   /**
    * sets the owner servlet object that instantiates an object of this class.
    *
    */
   public void setServlet(XMLGWforANYServlet _servlet) {
      servlet = _servlet;
   }

   /**
    * The JDBC connection to the database object.
    */
   public Connection connection;

   /**
    * The flag to let the thread run if it is true.
    */
   public boolean alive = true;

   /**
    * This method has to be implemented by the developer for the specific backend
database
    * to connect to it using JDBC and the function arguments.
    *
    * @return a boolean - true for a successful connection.
    *                   - false for an unsuccessful connection.
    *
    * @param dbserver the name or IP address of the database server.
    * @param dbport the port on which the JDBC server is running.
    * @param nameOfDataNase the name of the database instance.
    * @param userId the user id to connect to the database with.
    * @param password the password to connect to the database with.
```

```
     *
     */
    public abstract boolean connect(String dbserver, String dbport, String
nameOfDataBase, String userId, String password);

    /**
     * This method has to be implemented by the developer for the specific backend
database. It
     * allows one to add code that continually loops to pick up any records that may
be ready
     * for sending out from the database to the bridge. For e.g. in TSD, one has to
poll the Bridge_Trans
     * table as this table contains entries ready to be sent out. If no polling is
required for a
     * particular database, just do an empty implementation by adding in a pair of
empty braces, {},
     * and DO NOT call the start method in the class derived from the
XMLGWforANYServlet class.
     *
     */
    public abstract void run();


    /**
     * Replaces the first occurence of a substring in a string with another
substring.
     *
     * @param orig the the original string.
     * @param old the old substring to be replaced.
     * @param replacer the new substring to replace with.
     *
     * @return the modified string.
     *
     */
    public String replaceFirstString(String orig, String old, String replacer) {
        if(orig == null || old == null || replacer == null) {
            return orig;
        }

        int index = orig.indexOf(old);
        if(index != -1) { // old string found in orig string
            StringBuffer modifiedSB = new StringBuffer();
            modifiedSB.append(orig.substring(0, index));
            modifiedSB.append(replacer);
            modifiedSB.append(orig.substring(index+old.length()));
            return modifiedSB.toString();
        }
        else {          // old string not found in orig string
            return orig;
        }

    }


    /**
     * Replaces all occurences of a substring in a string with another substring.
     *
     * @param orig the the original string.
     * @param old the old substring to be replaced.
     * @param replacer the new substring to replace with.
     *
     * @return the modified string.
     *
```

```
 */
public String replaceString(String orig, String old, String replacer) {
   if(orig == null || old == null || replacer == null) {
      return orig;
   }

   while(orig.indexOf(old) != -1) {
      orig = replaceFirstString(orig, old, replacer);
   }

   return orig;
}

/**
 * set the database commit mode to ON via JDBC.
 *
 * @return a boolean indication whether or not the request was successful.
 *
 */
public boolean setAutoCommitOn() {
   try {
      connection.setAutoCommit(true);
   }
   catch(SQLException sqle) {
      servlet.log("problem in setting auto-commit to ON:" + sqle.toString());
      sqle.printStackTrace();
      return false;
   }

   return true;

}

/**
 * set the database commit mode to OFF via JDBC.
 *
 * @return a boolean indication whether or not the request was successful.
 */
public boolean setAutoCommitOff() {

   try {
      connection.setAutoCommit(false);
   }
   catch(SQLException sqle) {
      servlet.log("problem in setting auto-commit to OFF:" + sqle.toString());
      sqle.printStackTrace();
      return false;
   }

   return true;
}


/**
 * commits the transactions made thus far in the database via JDBC.
 *
 * @return a boolean indication whether or not the request was successful.
 */
public boolean commitDatabase() {
   try {
      connection.commit();
   }
   catch(SQLException sqle) {
```

```
            servlet.log("problem in committing database:" + sqle.toString());
            sqle.printStackTrace();
            return false;
        }

        return true;
    }


    /**
     * rolls back any of transactions made thus far in the database via JDBC.
     *
     */
    public void doRollback() {
        try {
            connection.rollback();
        }
        catch(SQLException sqle) {
            servlet.log("problem in rollback'ing database:" + sqle.toString());
            sqle.printStackTrace();
        }
    }


    /**
     * executes an INSERT, UPDATE, or DELETE query.
     *
     * @param query the query to execute.
     *
     * @return the number of rows in the database affected by this query.
     */
    public int executeUpdate(String query) {
        try {
            Statement stmt = connection.createStatement();
            int updatedRows = stmt.executeUpdate(query);
            //stmt.close();
            return updatedRows;
        }
        catch(SQLException ex) {
            servlet.log("problem in executing query, " + query + ": " + ex.toString());
            ex.printStackTrace();
            return -1;
        }
    }


    /**
     * executes an SQL statement that returns a single ResultSet. For example, a
SELECT query.
     *
     * @param query the query to execute.
     *
     * @return the ResultSet containing the queried(selected, for example) rows.
     */
    public ResultSet executeQuery(String query) {
        try {
            Statement stmt = connection.createStatement();
            ResultSet rs =  stmt.executeQuery(query);
            //stmt.close();
            return rs;
        }
        catch(SQLException ex) {
            servlet.log("problem in executing query, " + query + ": " + ex.toString());
```

```
            ex.printStackTrace();
            return null;
         }
      }


      /**
       * creates and returns a Prepared Statement that can be later executed
efficiently many
       * time by supplying the arguements.
       *
       * @param query the query to precompile in the database and keep ready for
execution.
       *
       * @return the precompiled statement, i.e., the PreparedStatement.
       */
      public PreparedStatement createPreparedStatement(String query) {
         try {
            PreparedStatement ps = connection.prepareStatement(query);
            return ps;
         }
         catch(Exception ex) {
            servlet.log("problem in creating prepared statement, " + query + ": " +
ex.toString());
            ex.printStackTrace();
            return null;
         }
      }

      /**
       *
       * This method executes a SELECT type of query and adds to the supplied hashtable
the resulting
       * data with the column names as the keys and the cell values as the values in
       * hashtable. It ASSUMES that only one row is returned from the database. If more
       * than one row is returned, then the very last one is the one that shows up in
       * the returned hashtable.
       *
       * @param query the string containing the SQL query.
       * @param hashtable the hashtable to which the resulting data is added.
       *
       * @return the hashtable to which the resulting data is added.
       */

      public Hashtable storeDataFromDatabase(String query, Hashtable hashTable) {
         try {
            ResultSet rs = executeQuery(query);

            ResultSetMetaData rsmd = rs.getMetaData();
            int totColumns = rsmd.getColumnCount();

            String columnName;
            String columnValue;

            while(rs.next()) {
               for(int i=1; i<=totColumns; i++) {
                  columnName  = rsmd.getColumnName(i);
                  columnValue = rs.getString(columnName);
                  hashTable.put(columnName, columnValue);
               }
            }
         }
         catch(Exception e) {
```

```
            log("problem in storing data using query, "+query+":"+e.toString());
            e.printStackTrace();
        }

        return hashTable;
    }

    /**
     *
     * This method takes a JDBC ResultSet and adds to the supplied hashtable the
result's
     * data with the column names as the keys and the cell values as the values in
     * hashtable. It ASSUMES that only one row is returned from the database. If more
     * than one row is returned, then the very last one is the one that shows up in
     * the returned hashtable.
     *
     * @param rs the ResultSet object from an executed query.
     * @param hashtable the hashtable to which the resulting data is added.
     *
     * @return the hashtable to which the resulting data is added.
     */

    public Hashtable storeDataFromDatabase(ResultSet rs, Hashtable hashTable) {
        try {
            ResultSetMetaData rsmd = rs.getMetaData();
            int totColumns = rsmd.getColumnCount();

            String columnName;
            String columnValue;

            while(rs.next()) {
                for(int i=1; i<=totColumns; i++) {
                    columnName  = rsmd.getColumnName(i);
                    columnValue = rs.getString(columnName);
                    if((columnName !=null) && (columnValue != null))  {
                        hashTable.put(columnName, columnValue);
                    }
                }
            }
        }
        catch(Exception e) {
            log("problem in storing data from database into hashtable:"+e.toString());
            e.printStackTrace();
        }

        return hashTable;
    }

    /**
     * disconnect from the database and close the connection.
     *
     */
    public void disconnect () {
        try {
            connection.close();
        }
        catch(SQLException ex) {
            servlet.log("problem in disconnecting: " + ex.toString());
            ex.printStackTrace();
        }
    }
```

```
    /**
     * a method that a developer should implement for the specific database
     * to block a set of new record numbers in the database system.
     *
     * @param range the range of ticket numbers to go until. Therefore, last ticket
number = nextTicketNumber+range-1
     *
     * @return a boolean indicating whether or not a transaction was successful
     * or not.
     *
     */
    abstract public boolean getRecordNumberStock(int range);


    /**
     * converts date from yyyy-mm-dd to mm/dd/yyyy format.
     *
     * @param value the date in yyy-mm-dd format.
     *
     * @return the date in mm/dd/yyyy format.
     *
     */
    public String reformatDate(String value) {

       if(value == null) return value;

       String yyyy = value.substring(0, 4);
       String mm   = value.substring(5, 7);
       String dd   = value.substring(8);

       String newDate = mm+"/"+dd+"/"+yyyy;
       return newDate;
    }


    /**
     * converts time from hh:mm AM|PM  to hh:mm:ss format.
     *
     * @param value the time in hh:mm AM|PM format.
     *
     * @return the time in hh:mm:ss format.
     *
     */
    public String reformatTime(String value) {

       if(value == null) return value;

       String hh = value.substring(0, 2);
       int hhInt = (new Integer(hh)).intValue();
       String mm   = value.substring(3, 5);
       String amOrPm   = value.substring(6);

       if(amOrPm.equals("PM"))
          hhInt += 12;  // convert to a 24 hr time cycle

       String newTime = hhInt+":"+mm+":"+"00";

       return newTime;
    }



    /**
```

```
     * removes dashes from the date.
     *
     * @param value the date WITH dashes in it.
     *
     * @return the the date WITHOUT dashes in it.
     *
     */
    public String removeDashFromDate(String value) {
       if(value == null)
          return value;

       int indexOfQuote = value.indexOf("-");
       if(indexOfQuote == -1)
          return value;
       else {
          StringBuffer sb = new StringBuffer();
          for(int i=0; i < value.length(); i++) {
             if(value.charAt(i) == '-') {

             }
             else {
                sb.append(value.charAt(i));
             }
          }
          return sb.toString();
       }
    }


    /**
     * this method can be overriden to send output and error messages to a desired
stream. By deafult it
     * prints it out to the standard output.
     *
     * @param msg the output or error message for logging purposes.
     *
     */
    public void log(String msg) {
       if(servlet != null) {
          servlet.log(msg);
       }
       else {
          System.out.println("*** Gateway: "+msg);
       }
    }
}
```

```
<?xml version="1.0"?><XML_BRIDGE_MAPPING name="tsd60_43" versionNumber="1_1"
sourceDTD="C:\XMLBridge\MappingEditor\samples\dtd\tsd60outbound.dtd"
targetDTD="C:\XMLBridge\MappingEditor\samples\dtd\tsd422inbound.dtd">
  <XML_BRIDGE_RULESET><XML_BRIDGE_RULES targetElement="GATEWAY.NEW"><RULE
type="1"><![CDATA[$(GATEWAY.NEW) =  $(SRC::GATEWAY.NEW)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES targetElement="GATEWAY.NAME"><RULE
type="1"><![CDATA[$(GATEWAY.NAME) =  $(SRC::GATEWAY.NAME)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES targetElement="GATEWAY.BRIDGE"><RULE
type="1"><![CDATA[$(GATEWAY.BRIDGE) =  $(SRC::GATEWAY.BRIDGE)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="GATEWAY.TIMESTAMP"><RULE type="1"><![CDATA[$(GATEWAY.TIMESTAMP) =
$(SRC::GATEWAY.TIMESTAMP) ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TRANSACTIONTYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.TRANSACTIONTYPE) =
( $(SRC::PROBLEMS.PROBLEM_CODE).equals("CLOSED") ||
 $(SRC::PROBLEMS.PROBLEM_CODE).equals("CLOSE_RESOLVED")) ? "3" : "1" ;
 ]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CREATECALL"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CREATECALL) =  $(MACRO::NEW)() ? "1" : "0"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CREATEPROBLEM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CREATEPROBLEM) =  $(MACRO::NEW)() ? "1" : "0"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CLOSEPROBLEM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CLOSEPROBLEM) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PICKUPDISPATCH"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PICKUPDISPATCH) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TRANSFERUSER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.TRANSFERUSER) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DONOTIFICATION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DONOTIFICATION) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.EXPLODEGROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.EXPLODEGROUP) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_ID"><RULE type="1"><![CDATA[String P =
$(MACRO::NEW)() ? $(EXTID::BRIDGE_TICKET_NO) :  $(SRC::PROBLEMS.BRIDGE_TICKET_NO)  ;
P =  $(MACRO::TRUNC15)(P) ;
P =  $(MACRO::PAD8)(P) ;
$(PROBLEM_CLOSURE.PROBLEM_ID) = P ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.MODIFY_DATETIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.MODIFY_DATETIME) =
"1";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_BEGIN_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_BEGIN_DATE) =  $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_BEGIN_DATE) :  "_" ;

]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_BEGIN_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_BEGIN_TIME) = $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_BEGIN_TIME) : "_" ;

]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_END_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_END_DATE) = $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_END_DATE) : "_" ;



]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
```

```
targetElement="PROBLEM_CLOSURE.SESSION_END_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_END_TIME) = $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_END_TIME) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLER_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALLER_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLER_NAME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALLER_NAME) =  $(MACRO::NEW)() ?
$(SRC::CALL.CALLER_NAME) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLER_PHONE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALLER_PHONE) =  $(MACRO::NEW)() ?
$(SRC::CALL.CALLER_PHONE) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.USER_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.USER_ID) =  $(SRC::GATEWAY.NAME)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALL_CODE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALL_CODE) =  $(SRC::SESION.CALL_CODE)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SEVERITY"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SEVERITY) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SEVERITY) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_CODE"><RULE type="1"><![CDATA[String P =
$(MACRO::OWNER)() ? $(SRC::PROBLEMS.PROBLEM_CODE) : "_" ;
if (P.equals("TRANSFERRED")) { P = "OPEN" ; }
else if (P.equals("OPEN")) { P = "TRANSFERRED" ; }
$(PROBLEM_CLOSURE.PROBLEM_CODE) = P;

]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_TYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PROBLEM_TYPE) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.PROBLEM_TYPE) : "_" ;



]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SYSTEM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SYSTEM) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SYSTEM) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.COMPONENT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.COMPONENT) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.COMPONENT) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ITEM"><RULE type="1"><![CDATA[$(PROBLEM_CLOSURE.ITEM)
= $(MACRO::OWNER)() ? $(SRC::PROBLEMS.ITEM) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.MODULE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.MODULE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.MODULE) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DESCRIPTION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DESCRIPTION) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.PROB_DESCRIPTION) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SERIAL_NUMBER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SERIAL_NUMBER) = $(MACRO::OWNER)() ?
$(SRC::SESION.SERIAL_NUMBER) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.INVENTORY_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.INVENTORY_ID) = $(MACRO::OWNER)() ?
$(SRC::SESION.INVENTORY_ID) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_RESULT"><RULE type="1"><![CDATA[String P =
($(MACRO::NEW)())  ? $(EXTID::BRIDGE_TICKET_NO) : "_"  ;
                        if  ($(MACRO::SWITCH)())  {                                P =
$(SRC::VENDOR_INFO.BRIDGE_TICKET_NO) ;                          }
                        else if  ($(MACRO::MATCH)())   {
```

```
P  =   $(SRC::PROBLEMS.PROBLEM_RESULT)  ;                          }
P =  $(MACRO::TRUNC15)(P) ;
P =  $(MACRO::PAD8)(P) ;
$(PROBLEM_CLOSURE.PROBLEM_RESULT) = P;
//OWNED BY OWNER AND USED TO SWITCH
OWNER]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TIME_SPENT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.TIME_SPENT) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DIAG_NODE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DIAG_NODE) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DIAG_NODE) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR1) =  $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR3) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR3) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR4) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR4) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT3) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT3) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT4) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT4) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_DATE1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_DATE1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_DATE1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_DATE2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_DATE2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_DATE2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_TIME1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_TIME1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_TIME1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_TIME2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_TIME2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_TIME2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR1) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_VCHR1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR2) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_VCHR2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR3) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_VCHR3) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT1) =
"1";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT2) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
```

```
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT3) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_INT3) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT4) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_INT4) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_DATE1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_DATE1) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_DATE1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_DATE2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_DATE2) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_DATE2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_TIME1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_TIME1) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_TIME1) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_TIME2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_TIME2) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_TIME2) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.LINE_NUMBER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.LINE_NUMBER) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFICATION_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFICATION_DATE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFICATION_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFICATION_TIME) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFICATION_TYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFICATION_TYPE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFY_CONTACT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFY_CONTACT) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SOLUTION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SOLUTION) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR5"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR5) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.GROUP_ID) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.GROUP_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.GROUP_ID) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.GROUP_ID) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.LOGGED_USER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.LOGGED_USER) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.LOGGED_USER) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ACTIVE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ACTIVE) =  $(MACRO::NEW)() ? "1" : "0"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR4) =  $(MACRO::NEW)() ?
$(SRC::CALL.CALLER_NAME) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD1) =  $(MACRO::OWNER)()  ?
$(SRC::PROBLEMS.VEND_FIELD1) :  "_"  ;
]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD2) =  $(MACRO::OWNER)()  ?
$(SRC::PROBLEMS.VEND_FIELD2) :  "_"  ;



]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD3"><RULE
```

```
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD3) =  $(MACRO::OWNER)()  ?
$(SRC::PROBLEMS.VEND_FIELD3)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD4) =  $(MACRO::OWNER)()  ?
$(SRC::PROBLEMS.VEND_FIELD4)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD5"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD5) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.VEND_FIELD5)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD6"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD6) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.VEND_FIELD6)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD7"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD7) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.VEND_FIELD7)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD8"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD8) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.VEND_FIELD8)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD9"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD9) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.VEND_FIELD9)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.VEND_FIELD10"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.VEND_FIELD10) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.VEND_FIELD10)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME1) =  $(MACRO::OWNER)()  ?
$(SRC::PROBLEMS.DATETIME1)  :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME2"><RULE
```

```
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME2) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME2) :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME3) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME3) :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME4) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME4) :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME5"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME5) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME5) :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME6"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME6) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME6) :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME7"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME7) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME7) :  "_"  ;




]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME8"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME8) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME8)  :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME9"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME9) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME9) :  "_"  ;


]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DATETIME10"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DATETIME10) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DATETIME10) :  "_"  ;

]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.BRIDGE_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.BRIDGE_ID) =  $(SRC::GATEWAY.BRIDGE)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.BRIDGE_TICKET_NO"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.BRIDGE_TICKET_NO) =
```

```
$(SRC::PROBLEMS.PROBLEM_ID) ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.INCOMING_FLAG"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.INCOMING_FLAG) = "1"; // UNLOCKS TICKET BY
CLEARING PROBLEMS.ACTIVE_WITH]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.BRIDGE_DESCRIPTION) :  "_"  ;



]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.RESOLVER_GROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.RESOLVER_GROUP) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.RESOLVER_GROUP) :  "_"  ;
]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.REPORTER_GROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.REPORTER_GROUP) =  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.REPORTER_GROUP) :  "_"  ;

]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.TEAM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.TEAM) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.CALLBACK_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.CALLBACK_DATE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.ORIG_TARGET_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.ORIG_TARGET_DATE)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.ORIG_TARGET_DATE) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.ORIG_TARGET_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.ORIG_TARGET_TIME)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.ORIG_TARGET_TIME) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.CURR_TARGET_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.CURR_TARGET_DATE)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CURR_TARGET_DATE) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.CURR_TARGET_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.CURR_TARGET_TIME)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CURR_TARGET_TIME) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.OCCURRED_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.OCCURRED_DATE)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.OCCURRED_DATE) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.OCCURRED_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.OCCURRED_TIME)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.OCCURRED_TIME) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.SOLVED_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.SOLVED_DATE)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SOLVED_DATE) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.SOLVED_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.SOLVED_TIME)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SOLVED_TIME) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.CAUSE_CHANGE_NO"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.CAUSE_CHANGE_NO)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CAUSE_CHANGE_NO) :  "_"  ;
]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.ORIGINAL_SEVERITY"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.ORIGINAL_SEVERITY)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.ORIGINAL_SEVERITY) :  "_"
```

```
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.DURATION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.DURATION)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DURATION) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.NODEID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.NODEID)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.NODEID) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.REASSIGNMENT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.REASSIGNMENT)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.REASSIGNMENT) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.PROBLEM_ABSTRACT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.PROBLEM_ABSTRACT)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.PROBLEM_ABSTRACT) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.PROBLEM_DUPLICATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.PROBLEM_DUPLICATE)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.PROBLEM_DUPLICATE) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.RCA_REQUIRED"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.RCA_REQUIRED)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.RCA_REQUIRED) :  "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.PREV_OCCURRED_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.PREV_OCCURRED_DATE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.RCA_TEXT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.RCA_TEXT)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.RCA_TEXT) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.CAUSE_CODE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.CAUSE_CODE)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CAUSE_CODE) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.REASON"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.REASON)=  $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.REASON) :  "_"  ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE2.PREV_OCCURRED_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE2.PREV_OCCURRED_TIME) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES targetElement="HISTORY"><RULE
type="3" source="WORK_HISTORY"><![CDATA[String P =  $(MACRO::NEW)() ?
$(EXTID::BRIDGE_TICKET_NO) : $(SRC::PROBLEMS.BRIDGE_TICKET_NO) ;
P =  $(MACRO::TRUNC15)(P) ;
P =  $(MACRO::PAD8)(P) ;
$(WORK_HISTORY.WORK_ID) = "_";
$(WORK_HISTORY.PROBLEM_ID) = P ;
$(WORK_HISTORY.USER_ID) =  $(SRC::GATEWAY.NAME) ;
$(WORK_HISTORY.WORK_BEGIN_DATE) =  $(SRC::WORK_HISTORY.WORK_BEGIN_DATE) ;
$(WORK_HISTORY.WORK_BEGIN_TIME) =  $(SRC::WORK_HISTORY.WORK_BEGIN_TIME) ;
$(WORK_HISTORY.WORK_END_DATE) =  $(SRC::WORK_HISTORY.WORK_END_DATE) ;
$(WORK_HISTORY.WORK_END_TIME) =  $(SRC::WORK_HISTORY.WORK_END_TIME) ;
$(WORK_HISTORY.DESCRIPTION) =  $(SRC::WORK_HISTORY.DESCRIPTION) ;
$(WORK_HISTORY.DESC_OVRFLW) =  $(SRC::WORK_HISTORY.DESC_OVRFLW) ;
$(WORK_HISTORY.MODIFY_DATETIME) = "1" ;
$(WORK_HISTORY.TIME_STAMP) =  $(SRC::WORK_HISTORY.TIME_STAMP) ;
$(WORK_HISTORY.ACTIVITY_ACTION_ID) =  $(SRC::WORK_HISTORY.ACTIVITY_ACTION_ID) ;

]]></RULE></XML_BRIDGE_RULES></XML_BRIDGE_RULESET>
<XML_BRIDGE_MACROS><MACRO name="NEW" type="BOOLEAN"><![CDATA[boolean retVal = false;
try {
  // add your own code here and set the variable
  // retVal to a boolean value ...
  // DO NOT add or change code above this line ...

  retVal = $(SRC::GATEWAY.NEW).equals("1") ;
```

```
  // DO NOT add or change code below this line ...
} catch( Throwable t ) {}
return retVal;
]]></MACRO><MACRO name="SWITCH" type="BOOLEAN"><![CDATA[boolean retVal = false;
try {
  // add your own code here and set the variable
  // retVal to a boolean value ...
  // DO NOT add or change code above this line ...

retVal = (("SWITCH" + ":" + $(SRC::PROBLEMS.PROBLEM_ID) + ":" +

$(SRC::VENDOR_INFO.BRIDGE_TICKET_NO)).equals( $(SRC::PROBLEMS.PROBLEM_RESULT) ));

  // DO NOT add or change code below this line ...
} catch( Throwable t ) {}
return retVal;
]]></MACRO><MACRO name="MATCH" type="BOOLEAN"><![CDATA[boolean retVal = false;
try {
  // add your own code here and set the variable
  // retVal to a boolean value ...
  // DO NOT add or change code above this line ...

  retVal = ($(SRC::PROBLEMS.PROBLEM_ID).equals( $(SRC::PROBLEMS.PROBLEM_RESULT) ));

  // DO NOT add or change code below this line ...
} catch( Throwable t ) {}
return retVal;
]]></MACRO><MACRO name="OWNER" type="BOOLEAN"><![CDATA[boolean retVal = false;
try {
  // add your own code here and set the variable
  // retVal to a boolean value ...
  // DO NOT add or change code above this line ...

  retVal = ($(MACRO::NEW)()||$(MACRO::MATCH)()||$(MACRO::SWITCH)());

  // DO NOT add or change code below this line ...
} catch( Throwable t ) {}
return retVal;
]]></MACRO><MACRO name="TRUNC15" type="STRING_ONE_PARAM"><![CDATA[String retVal =
"_";
String inputParam = input;
try {
  // add your own code here and set the variable
  // retVal to a String value ...
  // The variable inputParam references the input parameter
  // DO NOT add or change code above this line ...

  if (inputParam.length()>15) {
    inputParam = inputParam.substring(0,15);
  }
  retVal = inputParam ;

  // DO NOT add or change code below this line ...
} catch( Throwable t ) { retVal = "_"; }
return retVal;
]]></MACRO><MACRO name="PAD8" type="STRING_ONE_PARAM"><![CDATA[String retVal = "_";
String inputParam = input;
try {
  // add your own code here and set the variable
  // retVal to a String value ...
  // The variable inputParam references the input parameter
  // DO NOT add or change code above this line ...
```

```
  if (!inputParam.equals("_")) {
    int z = 8-inputParam.length() ;
    for (int i=0;i<z; i++){
      inputParam = "0"+inputParam ;
    }
  }
  retVal = inputParam ;
  // DO NOT add or change code below this line ...
} catch( Throwable t ) { retVal = "_"; }
return retVal;
]]></MACRO></XML_BRIDGE_MACROS></XML_BRIDGE_MAPPING>
```

```
<?xml version="1.0"?><XML_BRIDGE_MAPPING name="tsd43_60" versionNumber="1_1"
sourceDTD="C:\XMLBridge\MappingEditor\samples\dtd\tsd422outbound.dtd"
targetDTD="C:\XMLBridge\MappingEditor\samples\dtd\tsd60inbound.dtd">
  <XML_BRIDGE_RULESET><XML_BRIDGE_RULES targetElement="GATEWAY.NEW"><RULE
type="1"><![CDATA[$(GATEWAY.NEW) = $(SRC::GATEWAY.NEW)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES targetElement="GATEWAY.NAME"><RULE
type="1"><![CDATA[$(GATEWAY.NAME) =  $(SRC::GATEWAY.NAME)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES targetElement="GATEWAY.BRIDGE"><RULE
type="1"><![CDATA[$(GATEWAY.BRIDGE) =  $(SRC::GATEWAY.BRIDGE)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="GATEWAY.TIMESTAMP"><RULE type="1"><![CDATA[$(GATEWAY.TIMESTAMP) =
$(SRC::GATEWAY.TIMESTAMP) ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TRANSACTIONTYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.TRANSACTIONTYPE) =
( $(SRC::PROBLEMS.PROBLEM_CODE).equals("CLOSED") ||
 $(SRC::PROBLEMS.PROBLEM_CODE).equals("CLOSE RESOLVED") ) ?
"3" : "1" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CREATECALL"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CREATECALL) =  $(MACRO::NEW)()  ? "1" :
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CREATEPROBLEM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CREATEPROBLEM) =  $(MACRO::NEW)()  ? "1" :
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CLOSEPROBLEM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CLOSEPROBLEM) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PICKUPDISPATCH"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PICKUPDISPATCH) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TRANSFERUSER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.TRANSFERUSER) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DONOTIFICATION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DONOTIFICATION) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.EXPLODEGROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.EXPLODEGROUP) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.MODIFY_DATETIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.MODIFY_DATETIME) =
"1";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_BEGIN_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_BEGIN_DATE) =  $(MACRO::OWNER)()  ?
$(SRC::SESION.SESSION_BEGIN_DATE) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_BEGIN_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_BEGIN_TIME) = $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_BEGIN_TIME) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_END_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_END_DATE) = $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_END_DATE) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_END_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_END_TIME) = $(MACRO::OWNER)() ?
$(SRC::SESION.SESSION_END_TIME) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLER_NAME"><RULE type="1"><![CDATA[
$(PROBLEM_CLOSURE.CALLER_NAME) =  $(MACRO::NEW)()  ? $(SRC::CALL.CALLER_NAME) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLER_PHONE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALLER_PHONE) =  $(MACRO::NEW)()  ?
$(SRC::CALL.CALLER_PHONE) : "_" ; ]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
```

```
targetElement="PROBLEM_CLOSURE.USER_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.USER_ID) =    $(SRC::GATEWAY.NAME)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALL_CODE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALL_CODE) = $(SRC::SESION.CALL_CODE)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SEVERITY"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SEVERITY) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SEVERITY) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_CODE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PROBLEM_CODE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.PROBLEM_CODE) : "_" ;
if ($(PROBLEM_CLOSURE.PROBLEM_CODE).equals("TRANSFERRED")) {
 $(PROBLEM_CLOSURE.PROBLEM_CODE) = "OPEN" ;
}
else if ($(PROBLEM_CLOSURE.PROBLEM_CODE).equals("OPEN")) {
 $(PROBLEM_CLOSURE.PROBLEM_CODE) = "TRANSFERRED" ;
}
]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_TYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PROBLEM_TYPE) =  $(MACRO::OWNER)()  ?
$(SRC::PROBLEMS.PROBLEM_TYPE) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_DUPLICATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PROBLEM_DUPLICATE) =
$(SRC::PROBLEMS.PROBLEM_DUPLICATE) ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.OWNING_GROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.OWNING_GROUP) = $(SRC::PROBLEMS.GROUP_ID)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.OWNING_USER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.OWNING_USER) = $(SRC::PROBLEMS.USER_ID)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.STATUSBOARD"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.STATUSBOARD) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.REMINDER_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.REMINDER_DATE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="WORK_HISTORY"><RULE type="3" source="HISTORY"><![CDATA[String P =
$(MACRO::NEW)()  ? $(EXTID::BRIDGE_TICKET_NO) :
$(SRC::VENDOR_INFO.BRIDGE_TICKET_NO) ;
P =  $(MACRO::TRUNC15)(P) ;
P =  $(MACRO::PAD8)(P) ;
String Q = $(SRC::GATEWAY.NAME) ;
Q = $(MACRO::TRUNC6)(Q) ;
$(WORK_HISTORY.WORK_ID) = "_";
$(WORK_HISTORY.PROBLEM_ID) = P;
$(WORK_HISTORY.USER_ID) =  $(SRC::GATEWAY.NAME)  ;
$(WORK_HISTORY.WORK_BEGIN_DATE) =  $(SRC::WORK_HISTORY.WORK_BEGIN_DATE)  ;
$(WORK_HISTORY.WORK_BEGIN_TIME) =  $(SRC::WORK_HISTORY.WORK_BEGIN_TIME)  ;
$(WORK_HISTORY.WORK_END_DATE) =  $(SRC::WORK_HISTORY.WORK_END_DATE)  ;
$(WORK_HISTORY.WORK_END_TIME) =  $(SRC::WORK_HISTORY.WORK_END_TIME)  ;
$(WORK_HISTORY.DESCRIPTION) =  $(SRC::WORK_HISTORY.DESCRIPTION)  ;
$(WORK_HISTORY.DESC_OVRFLW) =  $(SRC::WORK_HISTORY.DESC_OVRFLW)  ;
$(WORK_HISTORY.SITE_ID) =  Q  ;
$(WORK_HISTORY.ACTIVITY_ACTION_ID) =  $(SRC::WORK_HISTORY.ACTIVITY_ACTION_ID)  ;
$(WORK_HISTORY.MODIFY_DATETIME) = "1" ;
$(WORK_HISTORY.TIME_STAMP) =  $(SRC::WORK_HISTORY.TIME_STAMP)  ;
$(WORK_HISTORY.PARENT_WORK_ID) = "_" ;



]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
```

```
targetElement="PROBLEM_CLOSURE.PROBLEM_ID"><RULE type="1"><![CDATA[String P =
$(MACRO::NEW)()  ? $(EXTID::BRIDGE_TICKET_NO) :
$(SRC::VENDOR_INFO.BRIDGE_TICKET_NO) ;
P =  $(MACRO::TRUNC15)(P) ;
P =  $(MACRO::PAD8)(P) ;
$(PROBLEM_CLOSURE.PROBLEM_ID) = P;

]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SESSION_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SESSION_ID) =
"1";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SYSTEM"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SYSTEM) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SYSTEM) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.COMPONENT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.COMPONENT) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.COMPONENT) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ITEM"><RULE type="1"><![CDATA[$(PROBLEM_CLOSURE.ITEM)
= $(MACRO::OWNER)() ?  $(SRC::PROBLEMS.ITEM) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.MODULE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.MODULE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.MODULE) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DESCRIPTION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DESCRIPTION) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DESCRIPTION) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SERIAL_NUMBER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SERIAL_NUMBER) = $(MACRO::OWNER)() ?
$(SRC::SESION.SERIAL_NUMBER) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.INVENTORY_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.INVENTORY_ID) = $(MACRO::OWNER)() ?
$(SRC::SESION.INVENTORY_ID) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_RESULT"><RULE type="1"><![CDATA[String P =
($(MACRO::NEW)())  ? $(EXTID::BRIDGE_TICKET_NO) : "_"  ;
                     if  ($(MACRO::SWITCH)())  {                                P =
$(SRC::VENDOR_INFO.BRIDGE_TICKET_NO) ;                    }
                    else if  ($(MACRO::MATCH)())   {
P  =   $(SRC::PROBLEMS.PROBLEM_RESULT)  ;                 }
P =  $(MACRO::TRUNC15)(P) ;
P =  $(MACRO::PAD8)(P) ;
$(PROBLEM_CLOSURE.PROBLEM_RESULT) = P;
//OWNED BY OWNER AND USED TO SWITCH
OWNER]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TIME_SPENT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.TIME_SPENT) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR4) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR4) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR4) =  $(MACRO::NEW)()  ?
$(SRC::CALL.CALLER_NAME) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT1) =
"1";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT2) =
"0";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ACTIVE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ACTIVE) =  $(MACRO::NEW)()  ? "1" : "0"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR5"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR5) = $(MACRO::OWNER)() ?
```

```
$(SRC::PROBLEMS.GROUP_ID) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.GROUP_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.GROUP_ID) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.GROUP_ID) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD1) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD1)  :
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.BRIDGE_TICKET_NO"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.BRIDGE_TICKET_NO) =  $(SRC::PROBLEMS.PROBLEM_ID)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.BRIDGE_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.BRIDGE_ID) =  $(SRC::GATEWAY.BRIDGE)
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.INCOMING_FLAG"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.INCOMING_FLAG) = "1"; // UNLOCKS TICKET BY
CLEARING PROBLEMS.ACTIVE_WITH]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ORIG_TARGET_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ORIG_TARGET_DATE) =  $(MACRO::NEW)()  ?
$(SRC::PROBLEMS.ORIG_TARGET_DATE) : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ORIG_TARGET_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ORIG_TARGET_TIME) =  $(MACRO::NEW)()  ?
$(SRC::PROBLEMS.ORIG_TARGET_TIME)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CURR_TARGET_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CURR_TARGET_DATE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CURR_TARGET_DATE)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CURR_TARGET_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CURR_TARGET_TIME) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CURR_TARGET_TIME)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.REASON"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.REASON) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.OCCURRED_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.OCCURRED_DATE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.OCCURRED_DATE)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.OCCURRED_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.OCCURRED_TIME) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.OCCURRED_TIME)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SOLVED_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SOLVED_DATE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SOLVED_DATE)  : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SOLVED_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SOLVED_TIME) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.SOLVED_TIME)  : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ACCEPT_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ACCEPT_DATE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ACCEPT_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ACCEPT_TIME) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RCA_TEXT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RCA_TEXT) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RCA_REQUIRED"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RCA_REQUIRED) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CAUSE_CODE"><RULE
```

```
type="1"><![CDATA[$(PROBLEM_CLOSURE.CAUSE_CODE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CAUSE_CODE)  : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.REPORTER_GROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.REPORTER_GROUP) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.REPORTER_GROUP)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.REPORTER_SITE_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.REPORTER_SITE_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RESOLVER_GROUP"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RESOLVER_GROUP) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.RESOLVER_GROUP)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RESOLVER_SITE_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RESOLVER_SITE_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CAUSE_CHANGE_NO"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CAUSE_CHANGE_NO) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CAUSE_CHANGE_NO)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ORIGINAL_SEVERITY"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ORIGINAL_SEVERITY) =  $(MACRO::NEW)()  ?
$(SRC::PROBLEMS.ORIGINAL_SEVERITY)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DURATION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DURATION) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DURATION)  : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.REASSIGNMENT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.REASSIGNMENT) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.REASSIGNMENT)  : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLBACK_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALLBACK_DATE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CALLBACK_DATE)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CALLBACK_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CALLBACK_TIME) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.CALLBACK_TIME)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.PROBLEM_ABSTRACT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.PROBLEM_ABSTRACT) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.PROBLEM_ABSTRACT)  : "_"
;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DIAG_NODE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DIAG_NODE) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.DIAG_NODE) : "_" ;]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR1)  : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR2)  : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_VCHR3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_VCHR3) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_VCHR3) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT1)  : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT3) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT3) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
```

```
targetElement="PROBLEM_CLOSURE.FLX_CAL_INT4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_INT4) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_INT4) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_DATE1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_DATE1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_DATE1) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_DATE2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_DATE2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_DATE2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_TIME1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_TIME1) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_TIME1) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_CAL_TIME2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_CAL_TIME2) = $(MACRO::OWNER)() ?
$(SRC::CALL.FLX_CAL_TIME2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR2) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_VCHR2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR3) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_VCHR3) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT3) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_INT3) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_INT4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_INT4) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_INT4) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_DATE1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_DATE1) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_DATE1) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_DATE2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_DATE2) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_DATE2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_TIME1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_TIME1) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_TIME1) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_TIME2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_TIME2) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_TIME2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RCV_GROUP_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RCV_GROUP_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.LINE_NUMBER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.LINE_NUMBER) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFICATION_DATE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFICATION_DATE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFICATION_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFICATION_TIME) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFICATION_TYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFICATION_TYPE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NOTIFY_CONTACT"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NOTIFY_CONTACT) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.NOTIFY_CONTACT) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.SOLUTION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.SOLUTION) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.AID_TYPE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.AID_TYPE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
```

```
targetElement="PROBLEM_CLOSURE.ANNOTATION_FILE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ANNOTATION_FILE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.CONTROL_TIME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.CONTROL_TIME) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.NODE_NAME"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.NODE_NAME) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.NODE_NAME)  : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.OS390_CHANGE"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.OS390_CHANGE) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.ALT_USER_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.ALT_USER_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD2) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD2)  :
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD3) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD3) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD4) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD4) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD5"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD5) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD5) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD6"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD6) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD6) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD7"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD7) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD7) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD8"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD8) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD8) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD9"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD9) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD9) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.VEND_FIELD10"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.VEND_FIELD10) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.VEND_FIELD10) :
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME1) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME1)  : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME2"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME2) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME2) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME3"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME3) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME3) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME4"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME4) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME4) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME5"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME5) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME5) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME6"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME6) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME6) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME7"><RULE
```

```
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME7) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME7) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME8"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME8) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME8) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME9"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME9) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME9) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DATETIME10"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DATETIME10) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.DATETIME10) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.BRIDGE_DESCRIPTION"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.BRIDGE_DESCRIPTION) = $(MACRO::OWNER)() ?
$(SRC::VENDOR_INFO.BRIDGE_DESCRIPTION)  :
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.TEAM"><RULE type="1"><![CDATA[$(PROBLEM_CLOSURE.TEAM)
= $(MACRO::OWNER)() ?  $(SRC::PROBLEMS.TEAM)  :
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.FLX_PRO_VCHR1"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.FLX_PRO_VCHR1) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.FLX_PRO_VCHR1) : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.LOGGED_USER"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.LOGGED_USER) = $(MACRO::OWNER)() ?
$(SRC::PROBLEMS.LOGGED_USER)  : "_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.REPORTER_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.REPORTER_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RESOLVER_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RESOLVER_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.EXT_HOST_ORIGIN"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.EXT_HOST_ORIGIN) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.RCV_SITE_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.RCV_SITE_ID) =
"_";]]></RULE></XML_BRIDGE_RULES><XML_BRIDGE_RULES
targetElement="PROBLEM_CLOSURE.DOCUMENT_ID"><RULE
type="1"><![CDATA[$(PROBLEM_CLOSURE.DOCUMENT_ID) =
"_";]]></RULE></XML_BRIDGE_RULES></XML_BRIDGE_RULESET>
<XML_BRIDGE_MACROS><MACRO name="NEW" type="BOOLEAN"><![CDATA[boolean retVal = false;

try {

  // add your own code here and set the variable

  // retVal to a boolean value ...

  // DO NOT add or change code above this line ...


  retVal = $(SRC::GATEWAY.NEW).equals("1") ;



  // DO NOT add or change code below this line ...

} catch( Throwable t ) {}

return retVal;]]></MACRO><MACRO name="SWITCH" type="BOOLEAN"><![CDATA[boolean retVal
= false;

try {
```

```
  // add your own code here and set the variable
  // retVal to a boolean value ...
  // DO NOT add or change code above this line ...


retVal = (("SWITCH" + ":" + $(SRC::PROBLEMS.PROBLEM_ID) + ":" +

$(SRC::VENDOR_INFO.BRIDGE_TICKET_NO)).equals( $(SRC::PROBLEMS.PROBLEM_RESULT) ));
  // DO NOT add or change code below this line ...
} catch( Throwable t ) {}
return retVal;
]]></MACRO><MACRO name="MATCH" type="BOOLEAN"><![CDATA[boolean retVal = false;

try {

  // add your own code here and set the variable

  // retVal to a boolean value ...

  // DO NOT add or change code above this line ...



retVal = ($(SRC::PROBLEMS.PROBLEM_ID).equals( $(SRC::PROBLEMS.PROBLEM_RESULT) ));



  // DO NOT add or change code below this line ...

} catch( Throwable t ) {}
```

```
return retVal;


]]></MACRO><MACRO name="OWNER" type="BOOLEAN"><![CDATA[boolean retVal = false;
try {
  // add your own code here and set the variable
  // retVal to a boolean value ...
  // DO NOT add or change code above this line ...
  retVal = ($(MACRO::NEW)()||$(MACRO::MATCH)()||$(MACRO::SWITCH)());

  // DO NOT add or change code below this line ...
} catch( Throwable t ) {}
return retVal;
]]></MACRO><MACRO name="TRUNC15" type="STRING_ONE_PARAM"><![CDATA[String retVal =
"_";
String inputParam = input;
try {
  // add your own code here and set the variable
  // retVal to a String value ...
  // The variable inputParam references the input parameter
  // DO NOT add or change code above this line ...

  if (inputParam.length()>15) {
    inputParam = inputParam.substring(0,15);
  }
  retVal = inputParam ;

  // DO NOT add or change code below this line ...
} catch( Throwable t ) { retVal = "_"; }
return retVal;
]]></MACRO><MACRO name="PAD8" type="STRING_ONE_PARAM"><![CDATA[String retVal = "_";
String inputParam = input;
try {
  // add your own code here and set the variable
  // retVal to a String value ...
  // The variable inputParam references the input parameter
  // DO NOT add or change code above this line ...

  if (!inputParam.equals("_")) {
    int z = 8-inputParam.length() ;
    for (int i=0;i<z; i++){
      inputParam = "0"+inputParam ;
    }
  }
  retVal = inputParam ;

  // DO NOT add or change code below this line ...
} catch( Throwable t ) { retVal = "_"; }
return retVal;
]]></MACRO><MACRO name="TRUNC6" type="STRING_ONE_PARAM"><![CDATA[String retVal =
"_";
String inputParam = input;
try {
  // add your own code here and set the variable
  // retVal to a String value ...
  // The variable inputParam references the input parameter
  // DO NOT add or change code above this line ...
  if (inputParam.length()>6) {
    inputParam = inputParam.substring(0,6);
  }
  retVal = inputParam ;
```

```
  // DO NOT add or change code below this line ...
} catch( Throwable t ) { retVal = "_"; }
return retVal;
]]></MACRO></XML_BRIDGE_MACROS></XML_BRIDGE_MAPPING>
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

/**
 * This class serves as the base class for classes that provide
 * the functionality to connect to specific databases using JDBC
 * and interacting with them for various purposes.
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

import java.sql.*;
import java.util.*;
import com.ibm.xmlbridge.*;

public class DB2Connector extends DBConnector {

    /**
     * the very next ticket number to use.
     */
    public int nextTicketNumber;

    /**
     * the offset for Universal Time.
     */
    public int adjustHours = 0;

    /**
     * This method is implemented by the developer for the specific backend database
     * to connect to it using JDBC.
     *
     * @return a boolean - true for a successful connection.
     *                   - false for an unsuccessful connection.
     *
     * @param dbserver the name or IP address of the database server.
     * @param dbport the port on which the JDBC server is running.
     * @param nameOfDataNase the name of the database instance.
     * @param userId th:qe user id to connect to the database with.
     * @param password the password to connect to the database with.
     *
     */
    public boolean connect(String dbserver, String dbport, String nameOfDataBase,
String userId, String password) {
        String url = "jdbc:db2://"+dbserver+":"+dbport+"/"+nameOfDataBase;

        // Register the DB2 driver to access DB2 databases
        try {
            // check for the existence of the required driver
            Class.forName("COM.ibm.db2.jdbc.net.DB2Driver");
        } catch (ClassNotFoundException cnfe) {
            log("problem in checking for DB2 jdbc driver: " + cnfe.toString());
            cnfe.printStackTrace();
            return false;
        }

        while (true) {
            try {
                connection = DriverManager.getConnection(url, userId, password);
```

```
                log("Connected to DB2.");

                // get the offset for Universal Time for the first time. Once the
thread starts we will
                // check it routinely. This initial one is needed as it is needed by
the parse methods
                // in the servlet to fill the $adjustHours in the queries it builds
                String qualifiedTableName =
((XMLGWforTSDServlet)servlet).tablesOwner + ".SYSTEM_PROFILE";
                ResultSet rs = executeQuery("SELECT CURRENT TIMEZONE FROM
"+qualifiedTableName);
                String UToffsetString;
                int UToffset = 0;
                try {
                    while (rs.next()) {
                        // get the relevant data out
                        UToffsetString = rs.getString(1);
                        if (UToffsetString != null)
                            UToffsetString = UToffsetString.trim();

                        // get the hours part by leaving out the mm and ss digits
                        UToffsetString = UToffsetString.substring(0,
UToffsetString.length()-4);
                        UToffset = (new Integer(UToffsetString)).intValue();
                        //log("UToffset="+UToffset);
                        adjustHours =
UToffset-((XMLGWforTSDServlet)servlet).GMToffset;
                        log("adjustHours="+adjustHours);
                    }
                } catch (Exception e) {
                    log("problem in getting results from SYSTEM_PROFILE table:" +
e.toString());
                    e.printStackTrace();
                }
                // get the offset for Universal Time

                return true;
            } catch (SQLException ex) {
                log("problem in connecting to DB2, db2jstrt may not have been
started or the database might be down or unreachable....will try again: " +
ex.toString());
                ex.printStackTrace();
            }

            // sleep for a minute and then loop again
            try {
                Thread.sleep(60*1000);
            } catch (Exception e) {
                log("problem in sleep while trying to connect with DB2: " +
e.toString());
                e.printStackTrace();
            }
        }
    }

    private String bridgeTicketNo = null;
    private String vendField1 = null;

    private String getRemoteTicketNumber(String problemId) {
        // the logic to detemine create or update, that is being sent out
        String qualifiedTableName;
        if (((XMLGWforTSDServlet)servlet).vendor_infoTablePresent) { // TSD4.3
            qualifiedTableName = ((XMLGWforTSDServlet)servlet).tablesOwner +
```

```
".VENDOR_INFO";
        } else {  // TSD6.0
            qualifiedTableName = ((XMLGWforTSDServlet)servlet).tablesOwner +
".PROBLEMS";
        }

        ResultSet rs = executeQuery("SELECT BRIDGE_TICKET_NO, VEND_FIELD1 FROM
"+qualifiedTableName+" WHERE PROBLEM_ID='"+problemId+"'");
        try {
            while (rs.next()) {
                bridgeTicketNo = rs.getString(1);

                // for the SWITCH ownership stuff
                vendField1 = rs.getString(2);
                // for the SWITCH ownership stuff

                return bridgeTicketNo;
            }
        } catch (Exception e) {
            log("problem in getting BRIDGE_TICKET_NO from "+qualifiedTableName+"
table:" + e.toString());
            e.printStackTrace();
        }

        return null;
    }


    /**
     * This method is implemented by the developer to add code that continually
loops in to pick
     * up any records that may be ready for sending out from the database to the
bridge. For e.g.
     * here in TSD, one has to poll the Bridge_Trans table as this table contains
entries ready to
     * be sent out. If no polling is required for a particular database, just do an
empty implementation
     * by adding in a pair of empty braces, {}, and DO NOT call the start method in
the class derived from
     * the XMLGWforANYServlet class.
     */
    public void run() {
        // assign parsed data structures from the outbound gtw file to local
variables first
        Vector preparedStatementVec    =
((XMLGWforTSDServlet)servlet).preparedStatementVec;
        Vector tableNameVec            = ((XMLGWforTSDServlet)servlet).tableNameVec;
        Vector dtdGroupNameVec         =
((XMLGWforTSDServlet)servlet).dtdGroupNameVec;
        Vector dtdFieldVecVec          =
((XMLGWforTSDServlet)servlet).dtdFieldVecVec;
        Vector defaultValueVecVec      =
((XMLGWforTSDServlet)servlet).defaultValueVecVec;

        // poll
        while (alive) {
            // routinely get the offset for Universal Time
            String qualifiedTableName = ((XMLGWforTSDServlet)servlet).tablesOwner +
".SYSTEM_PROFILE";
            ResultSet rs = executeQuery("SELECT CURRENT TIMEZONE FROM
"+qualifiedTableName);
            String UToffsetString;
            int UToffset = 0;
```

```
            try {
                while (rs.next()) {
                    // get the relevant data out
                    UToffsetString = rs.getString(1);
                    if (UToffsetString != null)
                        UToffsetString = UToffsetString.trim();

                    // get the hours part by leaving out the mm and ss digits
                    UToffsetString = UToffsetString.substring(0,
UToffsetString.length()-4);
                    UToffset = (new Integer(UToffsetString)).intValue();
                    //log("UToffset="+UToffset);
                    adjustHours = UToffset-((XMLGWforTSDServlet)servlet).GMToffset;
                    log("adjustHours="+adjustHours);
                }
            } catch (Exception e) {
                log("problem in getting results from SYSTEM_PROFILE table:" +
e.toString());
                e.printStackTrace();
            }
            // get the offset for Universal Time

            // when a CSR creates a ticket to be bridged in TSD, then
PROBLEMS.ACTIVE_WITH and BRIDGE_TRANS.BRIDGE_ID are both set to the id for that
bridge like XMLGATEWAY
            // look in the BRIDGE_TRANS table for any for BRIDGE_ID='XMLGATEWAY',
ensure every bridge had a unique id like ATTGEMS so that they only pick up their own
entries from the BRIDGE_TRANS table
            qualifiedTableName = ((XMLGWforTSDServlet)servlet).tablesOwner +
".BRIDGE_TRANS";
            rs = executeQuery("SELECT PROBLEM_ID FROM "+qualifiedTableName+" WHERE
BRIDGE_ID='"+((XMLGWforTSDServlet)servlet).bridgeid+"'");
            Vector problemIds = new Vector();
            try {
                String pid;
                while (rs.next()) {
                    pid = rs.getString(1);
                    if (pid != null)
                        pid = pid.trim();

                    problemIds.addElement(pid);
                }
            } catch (Exception e) {
                log("problem in getting results from BRIDGE_TRANS table:" +
e.toString());
                e.printStackTrace();
            }

            //  get all the fields for every problemId from PROBLEMS, etc. tables
            for (int i=0; i<problemIds.size(); i++) {
                String problemId = (String)problemIds.elementAt(i);

                // for this problem, find the last WORK_HISTORY record's WORK_ID
that contains the description of 'copy in remote system'
                qualifiedTableName = ((XMLGWforTSDServlet)servlet).tablesOwner +
".WORK_HISTORY";
                String message = "copy in
"+((XMLGWforTSDServlet)servlet).remoteGatewayName;
                rs = executeQuery("SELECT MAX(WORK_ID) FROM "+qualifiedTableName+"
WHERE PROBLEM_ID='"+problemId+"' AND DESCRIPTION='"+message+"'");
                int thresholdWorkIdInt = 0;
                try {
                    while (rs.next()) {
```

```
                        thresholdWorkIdInt = rs.getInt(1); // according to JDBC API,
this will return zero(0) incase the value is SQL NULL, so we are OK!
                    }
                } catch (Exception e) {
                    log("problem in getting the threshold WORK_ID from the
WORK_HISTORY table:" + e.toString());
                    e.printStackTrace();
                }
                // for this problem, find the last WORK_HISTORY record that contains
the description of 'copy in remote system'

                // get the BRIDGE_TICKET_NO and VEND_FIELD1 for this problem
                getRemoteTicketNumber(problemId);
                // get the BRIDGE_TICKET_NO and VEND_FIELD1 for this problem

                StringBuffer outboundXML = new StringBuffer();
                outboundXML.append("<?xml version=\"1.0\"
encoding=\"UTF-8\"?>\r\n");
                outboundXML.append("<TSD_OUTBOUND_TICKET>\r\n");
                outboundXML.append("    <GATEWAY>\r\n");
                if (bridgeTicketNo == null) { // new ticket!
                    outboundXML.append("        <GATEWAY.NEW>1</GATEWAY.NEW>\r\n");
                } else {
                    outboundXML.append("        <GATEWAY.NEW>0</GATEWAY.NEW>\r\n");
                }
                outboundXML.append("
<GATEWAY.NAME>"+((XMLGWforTSDServlet)servlet).gatewayName+"</GATEWAY.NAME>\r\n");
                outboundXML.append("
<GATEWAY.BRIDGE>"+((XMLGWforTSDServlet)servlet).bridgeid+"</GATEWAY.BRIDGE>\r\n");
                outboundXML.append("        <GATEWAY.TIMESTAMP>"+(new
java.util.Date()).toString()+"</GATEWAY.TIMESTAMP>\r\n");
                outboundXML.append("    </GATEWAY>\r\n");

                // get all the reqd. fields from all the tables
                int totTables = tableNameVec.size();
                for (int j=0; j<totTables; j++) {
                    // get the table name for this table
                    String tableName = (String)tableNameVec.elementAt(j);

                    // get the dtd group name for this table
                    String dtdGroupName = (String)dtdGroupNameVec.elementAt(j);

                    // get the tablefieldsVec for this table
                    Vector dtdFieldVec = (Vector)dtdFieldVecVec.elementAt(j);

                    // get the defaultValueVec for this table
                    Vector defaultValueVec =
(Vector)defaultValueVecVec.elementAt(j);

                    PreparedStatement ps =
(PreparedStatement)preparedStatementVec.elementAt(j);
                    try {
                        // dependent code -- can not be in the base class
                        ps.setString(1, problemId);
                        if (tableName.equals("WORK_HISTORY")) {
                           ps.setInt(2, thresholdWorkIdInt);
                        }
                        // dependent code -- can not be in the base class
                        rs = ps.executeQuery();
                    } catch (Exception e) {
                        log("problem in setting arguments to the prepared statement
or executing it, "+ps+":"+e.toString());
                        e.printStackTrace();
```

```
                    continue; // skip this table and query in the XML generation
                }

                // executed the query, now generate XML
                int totFields = dtdFieldVec.size();
                try {
                    String data;
                    int rows = 0;
                    while (rs.next()) {
                        rows++;
                        if ((j == 0) ||
(!(((String)dtdGroupNameVec.elementAt(j-1)).equals(dtdGroupName)))) { // write the
opening tag only if the first table or if the dtd group name is not the same as the
one before, i.e., already written
                            outboundXML.append("    <"+dtdGroupName+">\r\n");
                        }
                        for (int k=0; k<totFields; k++) {
                            data = rs.getString(k+1); // as getString() index
begins from 1

                            if (data != null && !data.equals("")) {
                                data = data.trim();
                            } else {  // use default
                                data = (String)defaultValueVec.elementAt(k);
                            }

                            outboundXML.append("
<"+(String)dtdFieldVec.elementAt(k)+">"+data+"</"+(String)dtdFieldVec.elementAt(k)+"
>\r\n");
                        } // end for k
                        if (((j+1) == totTables) ||
(!(((String)dtdGroupNameVec.elementAt(j+1)).equals(dtdGroupName)))) { // write the
closing tag only if the last table or if the dtd group name is not the same as the
one after, i.e., yet to be written
                            outboundXML.append("    </"+dtdGroupName+">\r\n");
                        }
                    }

                    if (rows==0) { // not a single row of data was obtained for
this last query, so manually fill the tags with the default values
                        if ((j == 0) ||
(!(((String)dtdGroupNameVec.elementAt(j-1)).equals(dtdGroupName)))) { // write the
opening tag only if the first table or if the dtd group name is not the same as the
one before, i.e., already written
                            outboundXML.append("    <"+dtdGroupName+">\r\n");
                        }
                        for (int k=0; k<totFields; k++) {
                            data = (String)defaultValueVec.elementAt(k);
                            outboundXML.append("
<"+(String)dtdFieldVec.elementAt(k)+">"+data+"</"+(String)dtdFieldVec.elementAt(k)+"
>\r\n");
                        } // end for k
                        if (((j+1) == totTables) ||
(!(((String)dtdGroupNameVec.elementAt(j+1)).equals(dtdGroupName)))) { // write the
closing tag only if the last table or if the dtd group name is not the same as the
one after, i.e., yet to be written
                            outboundXML.append("    </"+dtdGroupName+">\r\n");
                        }
                    }
                } catch (Exception e) {
                    log("problem in getting results and creating XML from table
"+tableName+" :"+e.toString());
                    e.printStackTrace();
```

```
                    }
                } // end for j
                outboundXML.append("</TSD_OUTBOUND_TICKET>");
                log(outboundXML.toString());

                // post the ticket to the bridge
                try {
                    boolean sent = false;
                    if (bridgeTicketNo == null) {  //  send new ticket!
                        sent = servlet.sendNewRecordToBridge(problemId,
outboundXML.toString());
                    } else {                            // send updated ticket
                        sent = servlet.sendUpdatedRecordToBridge(problemId,
bridgeTicketNo, outboundXML.toString());
                    }

                    if (sent) {
                        // delete from the BRIDGE_TRANS table
                        qualifiedTableName =
((XMLGWforTSDServlet)servlet).tablesOwner + ".BRIDGE_TRANS";
                        executeUpdate("DELETE FROM "+qualifiedTableName+" WHERE
PROBLEM_ID='"+problemId+"'");
                        // delete from the BRIDGE_TRANS table

                        // add in the additional WH record to specify that the copy
of just prior inserted records exists in the remote system
                        int modifyDateTime = 1; // hardcode
                        String workId = getWorkId(1);
                        qualifiedTableName =
((XMLGWforTSDServlet)servlet).tablesOwner + ".WORK_HISTORY";
                        String sqlString = "INSERT INTO " +qualifiedTableName+"
(WORK_ID, PROBLEM_ID, USER_ID, WORK_BEGIN_DATE, WORK_BEGIN_TIME, WORK_END_DATE,
WORK_END_TIME, DESCRIPTION, DESC_OVRFLW, MODIFY_DATETIME, TIME_STAMP) SELECT
"+workId+", '"+problemId+"', '"+((XMLGWforTSDServlet)servlet).gatewayName+"',
date(current timestamp) , time(current timestamp), date(current timestamp) ,
time(current timestamp), '"+message+"', ' ', "+modifyDateTime+", current timestamp
FROM "+((XMLGWforTSDServlet)servlet).tablesOwner+".SYSTEM_PROFILE";
                        log("sqlString="+sqlString+"\n\n");
                        if (executeUpdate(sqlString) < 1) {
                            log("ticket sent to Bridge but the 'copy in remote
system' WH record insertion failed!");
                        }
                        // add in the additional WH record to specify that the copy
of just prior inserted records exists in the remote system

                        // do the SWITCH of ownership stuff
                        if (vendField1 != null && vendField1.startsWith("SWITCH")) {
                            String newString =
vendField1.substring(vendField1.lastIndexOf(":") + 1);

                            if
(((XMLGWforTSDServlet)servlet).vendor_infoTablePresent) { // TSD4.3
                                qualifiedTableName =
((XMLGWforTSDServlet)servlet).tablesOwner + ".VENDOR_INFO";
                            } else {  // TSD6.0
                                qualifiedTableName =
((XMLGWforTSDServlet)servlet).tablesOwner + ".PROBLEMS";
                            }

                            sqlString = "UPDATE "+qualifiedTableName+" SET
VEND_FIELD1='"+newString+"' WHERE PROBLEM_ID='"+problemId+"'";
                            log("sqlString="+sqlString+"\n\n");
                            if (executeUpdate(sqlString) < 1) {
```

```
                                log("ticket sent to Bridge but the SWITCH string
update into "+qualifiedTableName+" failed!");
                            }
                        }
                        // do the SWITCH of ownership stuff

                        // unlock the ticket in the database, if an update.
Else(create) do this in the ACKTICKET
                        if (bridgeTicketNo != null) {
                            qualifiedTableName =
((XMLGWforTSDServlet)servlet).tablesOwner + ".PROBLEMS";
                            sqlString = "UPDATE "+qualifiedTableName+" SET
ACTIVE_WITH=null WHERE PROBLEM_ID='"+problemId+"'";
                            log("sqlString="+sqlString+"\n\n");
                            if (executeUpdate(sqlString) < 1) {
                                log("ticket sent to Bridge but the unlocking update
into PROBLEMS.ACTIVE_WITH failed!");
                            }
                        }
                        // unlock the ticket in the database, if an update.
Else(create) do this in the ACKTICKET

                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    log("problem in creating TSDOutboundTicket, ignoring ticket for
problemId="+problemId+":"+e.toString());
                    e.printStackTrace();
                    continue;
                }
            }// end for i

            // rest for a minute and then work again
            try {
                Thread.sleep(30*1000);
            } catch (InterruptedException ie) {
                log("problem in the sleeping: " + ie.toString());
                ie.printStackTrace();
            } // catch
        }
    } // end run

    /**
     * This method is implemented by the developer for the specific database
     * to block a set of new ticket numbers in the database system.
     *
     * @param range the range of ticket numbers to go until. Therefore, last ticket
number = nextTicketNumber+range-1
     *
     * @return a boolean indicating whether or not a transaction was successful
     * or not.
     *
     */
    public boolean getRecordNumberStock(int range) {
        String tableName = ((XMLGWforTSDServlet)servlet).tablesOwner + ".COUNTERS";
        if (!setAutoCommitOff()) return false;

        ((XMLGWforTSDServlet)servlet).log("range="+range);
        int updatedRows = executeUpdate("UPDATE "+tableName+" SET
NEXT_NUMBER=NEXT_NUMBER+"+range+" WHERE COUNTER_NAME='PROBLEM'");
        if (updatedRows < 1)
            return false;
```

```
        ResultSet rs = executeQuery("SELECT NEXT_NUMBER-"+range+" FROM "+tableName+"
WHERE COUNTER_NAME='PROBLEM'");
        if (rs == null)
            return false;

        try {
            while (rs.next()) {
                int ntn = rs.getInt(1);
                //int newNextTicketNumber = ntn + range;  why??

                if (!commitDatabase())
                    return false;

                nextTicketNumber = ntn;
                setAutoCommitOn();
            }
        } catch (Exception e) {
            ((XMLGWforTSDServlet)servlet).log("problem in reading/updating COUNTERS
table's NEXT_NUMBER:"+e.toString());
            e.printStackTrace();
            return false;
        }

        return true;
    }

    /**
     * Gets the next CLOSURE_ID from the COUNTERS table in TSD.
     *
     * @return a String contaning the next closure_id.
     */
    public String getClosureId() {
        String closureId = null;

        String tableName = ((XMLGWforTSDServlet)servlet).tablesOwner + ".COUNTERS";
        if (!setAutoCommitOff()) return null;


        int updatedRows = executeUpdate("UPDATE "+tableName+" SET
NEXT_NUMBER=NEXT_NUMBER+1 WHERE COUNTER_NAME='CLOSURE'");
        if (updatedRows < 1)
            return null;

        ResultSet rs = executeQuery("SELECT NEXT_NUMBER-1 FROM "+tableName+" WHERE
COUNTER_NAME='CLOSURE'");
        if (rs == null)
            return null;

        try {
            while (rs.next()) {
                int nn = rs.getInt(1);

                if (commitDatabase()) {
                    closureId = (new Integer(nn)).toString();
                } else
                    doRollback();

                setAutoCommitOn();
            }
        } catch (Exception e) {
            ((XMLGWforTSDServlet)servlet).log("problem in reading/updating COUNTERS
table's NEXT_NUMBER for CLOSURE_ID:"+e.toString());
            e.printStackTrace();
```

```
            return null;
        }

        return closureId;
    }

    /**
     * Gets the next CALL_ID from the COUNTERS table in TSD.
     *
     * @return a String contaning the next call_id.
     */
    public String getCallId() {
        String callId = null;

        String tableName = ((XMLGWforTSDServlet)servlet).tablesOwner + ".COUNTERS";
        if (!setAutoCommitOff()) return null;

        int updatedRows = executeUpdate("UPDATE "+tableName+" SET
NEXT_NUMBER=NEXT_NUMBER+1 WHERE COUNTER_NAME='CALL'");
        if (updatedRows < 1)
            return null;

        ResultSet rs = executeQuery("SELECT NEXT_NUMBER-1 FROM "+tableName+" WHERE
COUNTER_NAME='CALL'");
        if (rs == null)
            return null;

        try {
            while (rs.next()) {
                int nn = rs.getInt(1);

                if (commitDatabase()) {
                    callId = (new Integer(nn)).toString();
                } else
                    doRollback();

                setAutoCommitOn();
            }
        } catch (Exception e) {
            ((XMLGWforTSDServlet)servlet).log("problem in reading/updating COUNTERS
table's NEXT_NUMBER for callId:"+e.toString());
            e.printStackTrace();
            return null;
        }

        return callId;
    }

    /**
     * Gets the next WORK_ID from the COUNTERS table in TSD and reserves increment
number of WORK_ID's
     * in TSD.
     *
     * @param increment the increment by which the WORK_ID in the COUNTERS table is
incremented.
     *
     * @return a String contaning the next work_id.
     */
    public String getWorkId(int increment) {
        String workId = null;

        String tableName = ((XMLGWforTSDServlet)servlet).tablesOwner + ".COUNTERS";
        if (!setAutoCommitOff()) return null;
```

```
        int updatedRows = executeUpdate("UPDATE "+tableName+" SET
NEXT_NUMBER=NEXT_NUMBER+"+increment+" WHERE COUNTER_NAME='WORK HISTORY'");
        if (updatedRows < 1)
            return null;

        ResultSet rs = executeQuery("SELECT NEXT_NUMBER-"+increment+" FROM
"+tableName+" WHERE COUNTER_NAME='WORK HISTORY'");
        if (rs == null)
            return null;

        try {
            while (rs.next()) {
                int nn = rs.getInt(1);

                if (commitDatabase()) {
                    workId = (new Integer(nn)).toString();
                } else
                    doRollback();

                setAutoCommitOn();
            }
        } catch (Exception e) {
            ((XMLGWforTSDServlet)servlet).log("problem in reading/updating COUNTERS
table's NEXT_NUMBER for workId:"+e.toString());
            e.printStackTrace();
            return null;
        }

        return workId;
    }
}
```

```
/*
 * Copyright (C) International Business Machines, Inc.
 * All rights reserved.
 */

package com.ibm.xmlgateway;

/**
 * This class is derived from the base class XMLGWforANYServlet to
 * recieve data from the bridge and pass it onto the destination
 * database(TSD/DB2) and vice-versa. It is therefore a servlet that
 * runs under a servlet runner like IBM WebSphere2.0 and connects
 * on one side to the Bridge using a generic protocol and on the
 * other side to a backend database system like TSD/DB2.
 *
 * @author Vikas Krishna (vikas@us.ibm.com)
 * @version 1.0 Date:
 */

// import IBM XML for Java parser classes
import com.ibm.xml.parser.*;

// import W3C classes
import org.w3c.dom.*;

// import java classes
import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;

// import java servlet classes
import javax.servlet.*;
import javax.servlet.http.*;

// import xmlbridge classes
import com.ibm.xmlbridge.*;

public class XMLGWforTSDServlet extends XMLGWforANYServlet {

   /**
    * The id for the Bridge.
    */
   public String bridgeid;

   /**
    * The user id to prefix the table names with.
    */
   public String tablesOwner;

   /**
    * The offset from GMT of the location of the server containing the database.
    */
   public int GMToffset;

   /**
    * The boolean that is true if VENDOR_INFO table exists(pre TSD6.0, like TSD4.3)
or not(TSD6.0 and onwards). TRUE by default
    */
   public boolean vendor_infoTablePresent = true;

   /**
    * The siteID for this bridge in the TSD system
```

```
     */
    public String siteID = "SITEA";

    /**
     * The data in the database for a particular problem ticket that is used in place
of nulls in inbound updates
     */
    public Hashtable dataInDB;

    /**
     * This method is implemented by the developer to do  bunch of startup one time
stuff like loading
     * the parameters from the properties file xmlgwfordb.props, connecting to the
database, etc..
     *
     * @param xmlgatewayDirString the base directory for the XMLGateway files without
the ending '\' or '/'.
     * this string is picked up from the servlet's initializations parameters by the
init() method and passed
     * on to the this method automatically.
     *
     * @exception ServletException if the base XMLGateway dir is not specified or if
it is not a valid directory,
     * or also if a bad oubound DTD was specified.
     *
     */
    public void startUpSpecific(String xmlgatewayDirString) {
       // read in the start-up params
       Properties p = new Properties();
       try {
          p.load(new
FileInputStream(xmlgatewayDirString+File.separator+"supportfiles"+File.separator+"xm
lgwfortsd.props"));
       }
       catch(Exception e) {
          log("problem in reading loading xmlgwfortsd properties file: " +
e.toString());
          e.printStackTrace();
       }

       // load gatewayName from the properties file
       if(p.getProperty("gatewayName") == null) {
          log("No gatewayName in properties file.");
       }
       else
          gatewayName = p.getProperty("gatewayName");
       log("gatewayName = " + gatewayName);

       // load remoteGatewayName from the properties file
       if(p.getProperty("remoteGatewayName") == null) {
          log("No remoteGatewayName in properties file.");
       }
       else
          remoteGatewayName = p.getProperty("remoteGatewayName");
       log("remoteGatewayName = " + remoteGatewayName);

       // load tsdserver from the properties file
       String tsdserver;
       if(p.getProperty("tsdserver") == null) {
          log("No tsdserver in properties file.");
          // use local host IP as the default assuming DB2 is running on the same
host
          tsdserver = "127.0.0.1";
```

```
        }
        else
           tsdserver = p.getProperty("tsdserver");
        log("tsdserver = " + tsdserver);

        // load db2jstrtport from the properties file
        String db2jstrtport;
        if(p.getProperty("db2jstrtport") == null) {
           log("No db2jstrtport in properties file.");
           // use 6789 as the default as specified in UBD for NT
           db2jstrtport = "6789";
        }
        else
           db2jstrtport = p.getProperty("db2jstrtport");
        log("db2jstrtport = " + db2jstrtport);

        // load nameofdatabase from the properties file
        String nameofdatabase = null;
        if(p.getProperty("nameofdatabase") == null)
           log("No nameofdatabase in properties file.");
        else
           nameofdatabase = p.getProperty("nameofdatabase");
        log("nameofdatabase = " + nameofdatabase);

        // load userid from the properties file to connect to the db with
        String userid = null;
        if(p.getProperty("userid") == null)
           log("No userid in properties file.");
        else
           userid = p.getProperty("userid");
        log("userid = " + userid);

        // load password from the properties file to connect to the db with
        String password = null;
        if(p.getProperty("password") == null)
           log("No password in properties file.");
        else
           password = p.getProperty("password");
        log("password = " + password);

        // load table owner userid(e.g. EXAV in our install) from the properties file
        if(p.getProperty("tablesOwner") == null) {
           log("No tablesOwner in properties file.");
        }
        else
           tablesOwner = p.getProperty("tablesOwner");
        log("tablesOwner = " + tablesOwner);

        // load bridgeid(e.g. ATTGEMS for this bridge) from the properties file
        if(p.getProperty("bridgeid") == null) {
           log("No bridgeid in properties file.");
        }
        else
           bridgeid = p.getProperty("bridgeid");
        log("bridgeid = " + bridgeid);

        // load bridgeBaseURL from the properties file
        String bridgeBaseURL;
        if(p.getProperty("bridgeBaseURL") == null) {
           log("No bridgeBaseURL in properties file.");
           // use local host IP as the default assuming bridge servlet is running on
the same host
           bridgeBaseURL = "http://127.0.0.1/servlet/xmlbridge/gw";
```

```
      }
      else
         bridgeBaseURL = p.getProperty("bridgeBaseURL");
      log("bridgeBaseURL = " + bridgeBaseURL);

      // load GMToffset from the properties file
      if(p.getProperty("GMToffset") == null) {
         log("No GMToffset in properties file using -5.");
         // use local host IP as the default assuming bridge servlet is running on
the same host
         GMToffset = -5;
      }
      else
         GMToffset = (new Integer(p.getProperty("GMToffset"))).intValue();
      log("GMToffset = " + GMToffset);


      // load vendor_info_exists from the properties file, to determine TSD6.0 or
TSD4.3
      String vendor_info_exists = null;
      if(p.getProperty("vendor_info_exists") == null) {
         log("No vendor_info_exists in properties file");
      }
      else {
         vendor_info_exists = p.getProperty("vendor_info_exists");
         log("vendor_info_exists = " + vendor_info_exists);
         if(vendor_info_exists.equals("yes")) {  // TSD4.3
            vendor_infoTablePresent = true;
         }
         else {  // TSD6.0
            vendor_infoTablePresent = false;
         }
      }

      // load siteID(e.g. SITEA for this bridge) from the properties file
      if(p.getProperty("siteID") == null) {
         log("No siteID in properties file. Using default value of SITEA");
      }
      else
         siteID = p.getProperty("siteID");
      log("siteID = " + siteID);

      // create the database connection
      dbc = new DB2Connector();
      dbc.setServlet(this);
      dbc.connect(tsdserver, db2jstrtport, nameofdatabase, userid, password);

      // create the communicator to write to the Bridge
      try {
         xmlbc = new XMLBridgeCommunicator(logger, new URL(bridgeBaseURL),
remoteGatewayName);
      }
      catch(Exception e) {
         log("problem in creating XMLBridgeCommunicator:"+e.toString());
         e.printStackTrace();
      }

      // read and parse the pre-processed gtw files

parseInboundGTWFile(xmlgatewayDirString+File.separator+"gtws"+File.separator+"tsd_in
bound.gtw");

parseOutboundGTWFile(xmlgatewayDirString+File.separator+"gtws"+File.separator+"tsd_o
```

```
utbound.gtw");

      // ONLY if the DB NEEDS to be polled for outbound records and the run() method

      // in DBConnector HAS BEEN OVERRIDDEN, start the thread and let it run forever
      dbc.start();
   }

   Vector preparedStatementVec = new Vector();
   Vector tableNameVec = new Vector();
   Vector dtdGroupNameVec = new Vector();
   Vector dtdFieldVecVec = new Vector();
   Vector defaultValueVecVec = new Vector();

   private void parseOutboundGTWFile(String fileName) {
      try {
         FileReader f = new FileReader(fileName);
         BufferedReader br = new BufferedReader(f);

         // get the total number of tables to handle
         // pick up the first line - TABLES TO HANDLE@N
         String aLine = br.readLine();
         String numTablesString = (new StringVector(aLine, "@")).myElementAt(1);
         int numTables = Integer.parseInt(numTablesString);

         // process all tables
         for(int i=0; i<numTables; i++) {
            // skip the next comment line - #TABLE NAME@TOTAL FIELDS@FROM
CLAUSE@WHERE CLAUSE@DTD GROUP NAME
            aLine = br.readLine();

            // head line - pick up the TABLE NAME,TOTAL FIELDS,FROM CLAUSE,WHERE
CLAUSE,DTD GROUP NAME
            aLine = br.readLine();

            StringVector sv = new StringVector(aLine, "@");

            tableNameVec.addElement(sv.myElementAt(0));
            int totFields = Integer.parseInt(sv.myElementAt(1));
            String fromClause = sv.myElementAt(2);
            String whereClause = sv.myElementAt(3);
            dtdGroupNameVec.addElement(sv.myElementAt(4));

            // skip the next comment line - #DTDFIELD@SQL SUBSTRING@DEFAULT VALUE
            aLine = br.readLine();

            // now pick up the all the DTDFIELD,SQL SUBSTRING,DEFAULT VALUE
            // and build the SQL query
            Vector v1 = new Vector();  // storage for DTD.FIELD for XML generation
            Vector v2 = new Vector();  // storage for DEFAULT VALUE for XML
generation
            String sqlSubstring;
            StringBuffer queryStringSB = new StringBuffer();
            queryStringSB.append("SELECT ");
            for(int j=0; j<totFields; j++) {
               aLine = br.readLine();

               sv = new StringVector(aLine, "@");
               v1.addElement(sv.myElementAt(0)); // DTDFIELD
               sqlSubstring = sv.myElementAt(1); //
               v2.addElement(sv.myElementAt(2));

               queryStringSB.append(sqlSubstring);
```

```
                if(j != totFields-1) {
                   queryStringSB.append(", ");
                }
                else {
                   queryStringSB.append(" "); // as no comma needed after the last
field
                }
             }
             // append the from clause
             queryStringSB.append("FROM "+fromClause);
             queryStringSB.append(" ");
             // append the where clause
             queryStringSB.append("WHERE "+whereClause);

             // dependent code -- can not be in the base class
             String query = queryStringSB.toString();
             query = dbc.replaceString(query, "$tablesOwner", tablesOwner);
             query = dbc.replaceString(query, "$adjustHours", (new
Integer(((DB2Connector)dbc).adjustHours)).toString());
             // dependent code -- can not be in the base class

             dtdFieldVecVec.addElement(v1);
             defaultValueVecVec.addElement(v2);
             preparedStatementVec.addElement(dbc.createPreparedStatement(query));
          }
          br.close();
       }
       catch(Exception e) {
          log("problem in reading or parsing file "+fileName+":"+e.toString());
          e.printStackTrace();
       }
    }

    Vector queryVecIn = new Vector();
    Vector tableNameVecIn = new Vector();
    Vector tableFrequencyVecIn = new Vector();
    Vector preparedStatementVecIn = new Vector(); // stores queries to read ticket
data incase of null fields in inbound updates only

    Vector fieldVecVecIn = new Vector();
    Vector fieldTypeVecVecIn = new Vector();
    Vector defaultValueVecVecIn = new Vector();

    private void parseInboundGTWFile(String fileName) {
       try {
          FileReader f = new FileReader(fileName);
          BufferedReader br = new BufferedReader(f);

          // get the total number of tables to read data from incase of nulls in
updates request from Bridge
          // pick up the first line - TABLES TO READ DATA FOR THE NULLS IN UPDATES@N
          String aLine = br.readLine();
          String numTablesString = (new StringVector(aLine, "@")).myElementAt(1);
          int numTables = Integer.parseInt(numTablesString);

          // skip the next comment line - #QUERIES FOR READING THE DATA FOR NULLS IN
UPDATES
          aLine = br.readLine();

          // read the queries needed to store the data from all the numTables tables
          for(int i=0; i<numTables; i++) {
             // pick up the query
```

```
            aLine = br.readLine();

            // replace the $tablewOwner with the tablesOwner
            aLine = dbc.replaceString(aLine, "$tablesOwner", tablesOwner);

            // create and store a PreparedStatement for it
            preparedStatementVecIn.addElement(dbc.createPreparedStatement(aLine));
         }

         // get the total number of tables to insert into for creates and updates
         // pick up the line - TABLES TO INSERT OR UPDATE INTO@N
         aLine = br.readLine();
         numTablesString = (new StringVector(aLine, "@")).myElementAt(1);
         numTables = Integer.parseInt(numTablesString);

         // process each table
         for(int i=0; i<numTables; i++) {
            // skip the line - #TABLE NAME@TOTAL FIELDS@FREQUENCY
            aLine = br.readLine();

            // head line - pick up the TABLE NAME,TOTAL FIELDS,FREQUENCY
            aLine = br.readLine();

            StringVector sv = new StringVector(aLine, "@");

            String tableName = sv.myElementAt(0);
            tableNameVecIn.addElement(tableName);
            int totFields = Integer.parseInt(sv.myElementAt(1));
            tableFrequencyVecIn.addElement(sv.myElementAt(2));

            // skip the line FIELD@QUOTED OR NOT@DEFAULT
            aLine = br.readLine();

            // now pick up and store all the FIELD's and DEFAULT VALUE's
            // and build the SQL query
            Vector v1 = new Vector();  // for FIELD
            Vector v3 = new Vector();  // for FIELD TYPE
            Vector v2 = new Vector();  // for DEFAULT VALUE
            String field;
            String fieldType;
            StringBuffer queryStringSB = new StringBuffer();
            queryStringSB.append("INSERT INTO "+tablesOwner+"."+tableName+" VALUES
("");
            for(int j=0; j<totFields; j++) {

               // pick up the FIELD,QUOTED OR NOT,DEFAULT
               aLine = br.readLine();
               sv = new StringVector(aLine, "@");
               field = sv.myElementAt(0);
               v1.addElement(field);                  // store field
               fieldType = sv.myElementAt(1);         // quoted or not
               v3.addElement(fieldType);              // store field type
               v2.addElement(sv.myElementAt(2));      // store field's default value
               field = "'$"+field+"'";

               queryStringSB.append(field);

               if(j != totFields-1) {
                  queryStringSB.append(", "); // as no comma needed after last field
               }
            }
            queryStringSB.append(")");
```

```
            fieldVecVecIn.addElement(v1);
            fieldTypeVecVecIn.addElement(v3);
            defaultValueVecVecIn.addElement(v2);
            queryVecIn.addElement(queryStringSB.toString());
         }
         br.close();
      }
      catch(Exception e) {
         log("problem in reading parsing from file "+fileName+":"+e.toString());
         e.printStackTrace();
      }
   }

   /**
    * this method is implemented by the XMLGateway developer with record creation
code
    * specifically for the target database. The inbound data(a problem ticket for
example) is sent
    * in XML to this method which is then parsed to get the data for CREATING a new
record(a new
    * problem ticket for example) into the database.
    *
    * @param createRecordXML the string(SQL) containing the single quote.
    * @param remoteId the record Id in the remote database.
    * @param localId the record Id in the local database.
    *
    * @return a boolean indicating whether or not the request was successfully
completed.
    */
   public boolean createRecordInDB(String createRecordXML, String remoteId, String
localId) {
      // left pad with zeros and make localId 8 digits
      localId = modifyProblemId(localId);

      TagValueExtractor.setTXDocument(createRecordXML);
      // get the closureId for this create -- dependent code, can not be in base
class
      String closureId = ((DB2Connector)dbc).getClosureId();
      String callId = ((DB2Connector)dbc).getCallId();
      if(!vendor_infoTablePresent) { // TSD6.0 requires a prefix of SITEID- to the
callId
         callId = siteID+"-"+callId;
      }

      int totQueries = queryVecIn.size();
      // create new queries for each table with the data values filled in
      Vector newQueryVecIn = new Vector();

      // Vector to hold Vectors of values, needed in executing the PreparedStatement
incase of long string fields, for a table
      Vector preparedStatementValuesVecVec = new Vector();

      for(int i=0; i<totQueries; i++) {
         String query = (String)queryVecIn.elementAt(i);
         String tableName = (String)tableNameVecIn.elementAt(i);
         String tableFrequency = (String)tableFrequencyVecIn.elementAt(i);

         Vector fieldVec = (Vector)fieldVecVecIn.elementAt(i);
         Vector fieldTypeVec = (Vector)fieldTypeVecVecIn.elementAt(i);
         Vector defaultValueVec = (Vector)defaultValueVecVecIn.elementAt(i);

         // fill the data value in the query
         int totFields = fieldVec.size();
```

```
        String field;
        String fieldType;
        String fieldValue;
        String defaultValue;

        if(tableFrequency.equals("SINGLE")) { // handle single occurring XML groups
like <PROBLEM></PROBLEM> group
            // Vector to hold Strings of values, needed in executing the
PreparedStatement incase of long string fields, for this table
            Vector preparedStatementValuesVec = new Vector();

            // replace the '$CLOSURE_ID' in the query with the closureId for this
create
            query = dbc.replaceString(query, "'$CLOSURE_ID'", closureId);
            // replace the '$CALL_ID' in the query with the callId for this create
            query = dbc.replaceString(query, "'$CALL_ID'", "'"+callId+"'");

            // replace the other '$macro' in the query with the corresponding values
for this create
            for(int j=0; j<totFields; j++) {
               field = (String)fieldVec.elementAt(j);
               fieldType = (String)fieldTypeVec.elementAt(j);
               // get the value from the incoming XML
               fieldValue =
escapeSingleQuote(TagValueExtractor.getTagFirstValue(tableName+"."+field));
               if(fieldValue.equals("_")) { // mapping sent null("_")
                  // supply a default specified for this fields in the inbound gtw
file
                  fieldValue = (String)defaultValueVec.elementAt(j);
               }

               // replace the '$macro' for this field in the query
               if(fieldValue.equals("_")) {  // still no value found so lets put
null in the database instead of "_", as no info.(null) is better than bad info.("_")
                  fieldValue = "null";
                  query = dbc.replaceString(query, ("'$"+field+"'"), fieldValue);
               }
               else {                    // some value found in incoming XML or from
the inbound gtw file, so lets use that
                  if(fieldType.startsWith("QUOTED")) {
                     if(fieldType.endsWith("LONGSTRING")) { // if a field happens to
exceed 255 chars, then we have to use a PreparedStatement to execute the insert
query, so mark it with a ? and collect it value to be used later in the setString()
method on the PreparedStatement
                        query = dbc.replaceString(query, ("'$"+field+"'"), (" ? "));
                        preparedStatementValuesVec.addElement(fieldValue);
                     }
                     else
                        query = dbc.replaceString(query, ("'$"+field+"'"),
("'"+fieldValue+"'"));
                  }
                  else {
                     query = dbc.replaceString(query, ("'$"+field+"'"), fieldValue);
                  }
               }
            }
            newQueryVecIn.addElement(query);
            preparedStatementValuesVecVec.addElement(preparedStatementValuesVec);
         }
         else {                                    // handle multiple occurring XML
groups like <HISTORY></HISTORY> group
            // create an array of totFields Vectors. Each Vector will hold the
```

```
multiple values for each XML tag
            Vector vecArray[] = new Vector[totFields];
            for(int j=0; j<totFields; j++) {
               vecArray[j] = new Vector();
            }

            // get the multiple values for each field
            for(int j=0; j<totFields; j++) {
               field = (String)fieldVec.elementAt(j);
               // get the values from the incoming XML
               vecArray[j] = TagValueExtractor.getTagValues(tableName+"."+field);
            }

            // find out how many <HISTORY> groups there are
            // as we have to do that many inserts in WORK_HISTORY,
            // hence that many WORK_IDs needed
            // dependent code, can not be in base class
            int howManyWHs = (vecArray[0]).size(); // the size of the Vectors
holding data for each WORK_HISTORY tag, i.e., the number of WORK_HISTORY records to
handle
            String startWorkIdString = ((DB2Connector)dbc).getWorkId(howManyWHs);
            int startWorkIdInt = (new Integer(startWorkIdString)).intValue();
            int workIdInt = startWorkIdInt;
            for(int j=0; j<howManyWHs; j++) {
               // Vector to hold Strings of values, needed in executing the
PreparedStatement incase of long string fields, for this table
               Vector preparedStatementValuesVec = new Vector();

               // first replace the '$WORK_ID' in the query with the workId for this
work_history record
               String workIdString = (new Integer(workIdInt)).toString();
               String aWHQuery;
               aWHQuery = dbc.replaceString(query, "'$WORK_ID'", workIdString);
               workIdInt++;

               // now replace the other '$macro' in the query with the corresponding
values for this create, note '$WORK_ID' has already been replaced with a value so no
harm even if we start from zero in the for loop. This was because WORK_ID could be
anyhwere in the tag order.
               for(int k=0; k<totFields; k++) {
                  field = (String)fieldVec.elementAt(k);
                  fieldType = (String)fieldTypeVec.elementAt(k);
                  // get the value from Vector created from the incoming XML
                  fieldValue =
escapeSingleQuote((String)(vecArray[k].elementAt(j)));
                  if(fieldValue.equals("_")) { // mapping sent null("_")
                     // supply a default specified for this fields in the inbound
gtw file
                     fieldValue = (String)defaultValueVec.elementAt(k);
                  }

                  // replace the '$macro' for this field in the query
                  if(fieldValue.equals("_")) {  // still no value found so lets put
null in the database instead of "_", as no info.(null) is better than bad info.("_")
                     fieldValue = "null";
                     aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
fieldValue);
                  }
                  else {                  // some value found in incoming XML or
from the inbound gtw file, so lets use that
                     if(fieldType.startsWith("QUOTED")) {
                        if(fieldType.endsWith("LONGSTRING")) { // if a field happens
to exceed 255 chars, then we have to use a PreparedStatement to execute the insert
```

```
query, so mark it with a ? and collect it value to be used later in the setString()
method on the PreparedStatement
                        aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
(" ? "));
                        preparedStatementValuesVec.addElement(fieldValue);
                     }
                     else
                        aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
("'"+fieldValue+"'"));
                  }
                  else {
                     aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
fieldValue);
                  }
               }
            }
            newQueryVecIn.addElement(aWHQuery);
            preparedStatementValuesVecVec.addElement(preparedStatementValuesVec);
         }
      }
   }

   // now exceute all queries atomically
   // set auto commit to OFF
   if(!dbc.setAutoCommitOff()) {
      log("problem in turning of auto commit, hence aborting create ticket.");
      return false;
   }

   int totNewQueries = newQueryVecIn.size();
   String sqlString;
   Vector preparedStatementValuesVec;
   for(int i=0; i<totNewQueries; i++) {
      sqlString = (String)newQueryVecIn.elementAt(i);
      log("sqlString="+sqlString);
      preparedStatementValuesVec =
(Vector)preparedStatementValuesVecVec.elementAt(i);
      int psvvSize = preparedStatementValuesVec.size();
      if(psvvSize > 0) { // this one is to be executed as a PreparedStatement!
         PreparedStatement ps = dbc.createPreparedStatement(sqlString);
         try {
            for(int j=0; j<psvvSize; j++) {
               String val = (String)preparedStatementValuesVec.elementAt(j);
               log("value="+val);
               ps.setString(j+1, val);
            }
            ps.executeQuery();
         }
         catch(Exception e) {
            log("problem in setting arguments to or executing the prepared
statement, "+ps+":"+e.toString());
            e.printStackTrace();
            dbc.doRollback();
            dbc.setAutoCommitOn();
            return false;
         }
      }
      else {            // this one can be executed directly
         if(dbc.executeUpdate(sqlString) < 1) {
            dbc.doRollback();
            dbc.setAutoCommitOn();
            return false;
         }
```

```
            }
        }

        // hardcode
        int modifyDateTime = 1;
        String message = "copy in "+remoteGatewayName;
        String workId = ((DB2Connector)dbc).getWorkId(1);

        // add in the additional WH record to specify that the copy of just prior
inserted records exists in the remote system
        String tableName = tablesOwner + ".WORK_HISTORY";
        sqlString = "INSERT INTO "+tableName+" (WORK_ID, PROBLEM_ID, USER_ID,
WORK_BEGIN_DATE, WORK_BEGIN_TIME, WORK_END_DATE, WORK_END_TIME, DESCRIPTION,
DESC_OVRFLW, MODIFY_DATETIME, TIME_STAMP) SELECT "+workId+", '"+localId+"',
'"+gatewayName+"', date(current timestamp) , time(current timestamp), date(current
timestamp) , time(current timestamp), '"+message+"', ' ', "+modifyDateTime+",
current timestamp FROM "+tablesOwner+".SYSTEM_PROFILE";
        log("sqlString="+sqlString+"\n\n");
        if(dbc.executeUpdate(sqlString) < 1) {
            dbc.doRollback();
            dbc.setAutoCommitOn();
            return false;
        }

        // carry out the insertion into the BRIDGE_CLOSURE table
        tableName = tablesOwner + ".BRIDGE_CLOSURE";
        sqlString = "INSERT INTO " +tableName+" (CLOSURE_ID) VALUES ("+closureId+")";
        log("sqlString="+sqlString+"\n\n");
        if(dbc.executeUpdate(sqlString) < 1) {
            dbc.doRollback();
            dbc.setAutoCommitOn();
            return false;
        }

        // ALL DONE! - commit and set auto commit back to ON
        dbc.commitDatabase();
        dbc.setAutoCommitOn();

        return true;
    }

    /**
     * this method is implemented by the XMLGateway developer with record updating
code
     * specifically for the target database. The inbound data(a problem ticket for
example) is sent
     * in XML to this method which is then parsed to get the data for UPDATING a new
record(a new
     * problem ticket for example) into the database.
     *
     * @param updateRecordXML the string(SQL) containing the single quote.
     * @param remoteId the record Id in the remote database.
     * @param localId the record Id in the local database.
     *
     * @return a boolean indicating whether or not the request was successfully
completed.
     */
    public boolean updateRecordInDB(String updateRecordXML, String remoteId, String
localId) {
        // left pad with zeros and make localId 8 digits
        localId = modifyProblemId(localId);

        if(!vendor_infoTablePresent) { // TSD6.0 requires a prefix of SITEID- to the
```

```
problem id
            localId = siteID+"-"+localId;
        }

        // set auto commit to OFF
        if(!dbc.setAutoCommitOff()) {
            log("problem in turning of auto commit, hence aborting update ticket.");
            return false;
        }

        // first check if the ticket is locked. If not lock it else reject it.
        String qualifiedTableName = tablesOwner + ".PROBLEMS";
        ResultSet rs = dbc.executeQuery("SELECT ACTIVE_WITH FROM
"+qualifiedTableName+" WHERE PROBLEM_ID='"+localId+"'");
        String activeWith = null;
        try {
            while(rs.next()) {
                activeWith = rs.getString(1);
            }
        }
        catch(Exception e) {
            log("problem in getting ACTIVE_WITH from PROBLEMS table:" + e.toString());
            dbc.doRollback();
            dbc.setAutoCommitOn();
            e.printStackTrace();
        }

        if(activeWith != null) {  // locked already, reject the update
            log("ticket, problem_id="+localId+", locked! Rejecting update to it.");
            return false;
        }
        else {                          // lock it with the remote gateway's id
            String sqlString = "UPDATE "+qualifiedTableName+" SET
ACTIVE_WITH='"+remoteGatewayName+"' WHERE PROBLEM_ID='"+localId+"'";
            log("sqlString="+sqlString);
            if(dbc.executeUpdate(sqlString) < 1) {
                log("problem in locking the ticket, problem_id="+localId+". It might not
exist in the database! Rejecting update to this ticket.");
                dbc.doRollback();
                dbc.setAutoCommitOn();
                return false;
            }
        }
        // first check if the ticket is locked. If not lock it else reject it.

        // get the ticket view for this ticket from the database
        Hashtable ticketView = new Hashtable(); // contains some of the data for the
ticket with problemId=localId in the db before this update
        int totPreparedStatements = preparedStatementVecIn.size();
        for(int i=0; i<totPreparedStatements; i++) {
            PreparedStatement ps =
(PreparedStatement)preparedStatementVecIn.elementAt(i);
            try {
                // dependent code -- can not be in the base class
                ps.setString(1, localId);
                // dependent code -- can not be in the base class
                rs = ps.executeQuery();
                dbc.storeDataFromDatabase(rs, ticketView);
            }
            catch(Exception e) {
                log("problem in setting arguments to the prepared statement,
"+ps+":"+e.toString());
                e.printStackTrace();
```

```
            continue; // skip this prepared statement
         }
      }

      TagValueExtractor.setTXDocument(updateRecordXML);
      // get the closureId for this update -- dependent code, can not be in base
class
      String closureId = ((DB2Connector)dbc).getClosureId();

      int totQueries = queryVecIn.size();

      // create new queries for each table with the data values filled in
      Vector newQueryVecIn = new Vector();

      // Vector to hold Strings of values, needed in executing the PreparedStatement
incase of long string fields, for this table
      Vector preparedStatementValuesVecVec = new Vector();

      for(int i=0; i<totQueries; i++) {
         String query = (String)queryVecIn.elementAt(i);
         String tableName = (String)tableNameVecIn.elementAt(i);
         String tableFrequency = (String)tableFrequencyVecIn.elementAt(i);

         Vector fieldVec = (Vector)fieldVecVecIn.elementAt(i);
         Vector fieldTypeVec = (Vector)fieldTypeVecVecIn.elementAt(i);
         Vector defaultValueVec = (Vector)defaultValueVecVecIn.elementAt(i);

         // fill the data value in the query
         int totFields = fieldVec.size();

         String field;
         String fieldType;
         String fieldValue;
         String defaultValue;

         if(tableFrequency.equals("SINGLE")) { // handle single occurring XML groups
like <PROBLEM></PROBLEM> group
            // Vector to hold Strings of values, needed in executing the
PreparedStatement incase of long string fields, for this table
            Vector preparedStatementValuesVec = new Vector();

            // replace the '$CLOSURE_ID' in the query with the closureId for this
update(for PROBLEM_CLOSURE and PROBLEM_CLOSURE2 tables)
            query = dbc.replaceString(query, "'$CLOSURE_ID'", closureId);

            // replace the other '$macro' in the query with the corresponding values
for this update
            for(int j=0; j<totFields; j++) {
               field = (String)fieldVec.elementAt(j);
               fieldType = (String)fieldTypeVec.elementAt(j);
               // get the value from the incoming XML
               fieldValue =
escapeSingleQuote(TagValueExtractor.getTagFirstValue(tableName+"."+field));
               if(fieldValue.equals("_")) { // mapping sent null("_")
                  // supply the value read in from the database for this field
                  fieldValue = escapeSingleQuote((String)ticketView.get(field));
                  // if null obtained from ticket view, supply a default specified
for this fields in the inbound gtw file
                  if(fieldValue == null) {
                     fieldValue = (String)defaultValueVec.elementAt(j);
                  }
               }
```

```
                // replace the '$macro' for this field in the query
                if(fieldValue.equals("_")) {  // still no value found so lets put
null in the database instead of "_", as no info.(null) is better than bad info.("_")
                   fieldValue = "null";
                   query = dbc.replaceString(query, ("'$"+field+"'"), fieldValue);
                }
                else {                    // some value found in incoming XML or from
the inbound gtw file, so lets use that
                   if(fieldType.startsWith("QUOTED")) {
                      if(fieldType.endsWith("LONGSTRING")) { // if a field happens to
exceed 255 chars, then we have to use a PreparedStatement to execute the insert
query, so mark it with a ? and collect it value to be used later in the setString()
method on the PreparedStatement
                         query = dbc.replaceString(query, ("'$"+field+"'"), (" ? "));
                         preparedStatementValuesVec.addElement(fieldValue);
                      }
                      else
                         query = dbc.replaceString(query, ("'$"+field+"'"),
("'"+fieldValue+"'"));
                   }
                   else {
                      query = dbc.replaceString(query, ("'$"+field+"'"), fieldValue);
                   }
                }
             }
             newQueryVecIn.addElement(query);
             preparedStatementValuesVecVec.addElement(preparedStatementValuesVec);
          }
          else {                                  // handle multiple occurring XML
groups like <HISTORY></HISTORY> group
             // create an array of totFields Vectors. Each Vector will hold the
multiple values for each XML tag
             Vector vecArray[] = new Vector[totFields];
             for(int j=0; j<totFields; j++) {
                vecArray[j] = new Vector();
             }

             // get the multiple values for each field
             for(int j=0; j<totFields; j++) {
                field = (String)fieldVec.elementAt(j);
                // get the values from the incoming XML but for the work_id's as we
are starting from j=1
                vecArray[j] = TagValueExtractor.getTagValues(tableName+"."+field);
             }

             // find out how many <HISTORY> groups there are
             // as we have to do that many inserts in WORK_HISTORY,
             // hence that many WORK_IDs needed
             // dependent code, can not be in base class
             int howManyWHs = (vecArray[0]).size(); // the size of the Vectors
holding data for each WORK_HISTORY tag, i.e., the number of WORK_HISTORY records to
handle
             String startWorkIdString = ((DB2Connector)dbc).getWorkId(howManyWHs);
             int startWorkIdInt = (new Integer(startWorkIdString)).intValue();
             int workIdInt = startWorkIdInt;
             // create a Vector of the work_ids for this update

             for(int j=0; j<howManyWHs; j++) {
                // Vector to hold Strings of values, needed in executing the
PreparedStatement incase of long string fields, for this table
                Vector preparedStatementValuesVec = new Vector();

                // first replace the '$WORK_ID' in the query with the workId for this
```

```
work_history record - for the WORK_HISTORY table
                String workIdString = (new Integer(workIdInt)).toString();
                String aWHQuery;
                aWHQuery = dbc.replaceString(query, "'$WORK_ID'", workIdString);
                workIdInt++;

                // now replace the other '$macro' in the query with the corresponding
values for this update
                for(int k=0; k<totFields; k++) {
                   field = (String)fieldVec.elementAt(k);
                   fieldType = (String)fieldTypeVec.elementAt(k);
                   // get the value from Vector created from the incoming XML
                   fieldValue =
escapeSingleQuote((String)(vecArray[k].elementAt(j)));
                   if(fieldValue.equals("_")) { // "_" is mapping's default for null
                      // supply a default specified for this fields in the inbound
gtw file
                      fieldValue = (String)defaultValueVec.elementAt(k);
                   }

                   // replace the '$macro' for this field in the query
                   if(fieldValue.equals("_")) {  // still no value found so lets put
null in the database instead of "_", as no info.(null) is better than bad info.("_")
                      fieldValue = "null";
                      aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
fieldValue);
                   }
                   else {                    // some value found in incoming XML or
from the inbound gtw file, so lets use that
                      if(fieldType.startsWith("QUOTED")) {
                         if(fieldType.endsWith("LONGSTRING")) { // if a field happens
to exceed 255 chars, then we have to use a PreparedStatement to execute the insert
query, so mark it with a ? and collect it value to be used later in the setString()
method on the PreparedStatement
                            aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
(" ? "));
                            preparedStatementValuesVec.addElement(fieldValue);
                         }
                         else
                            aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
("'"+fieldValue+"'"));
                      }
                      else {
                         aWHQuery = dbc.replaceString(aWHQuery, ("'$"+field+"'"),
fieldValue);
                      }
                   }
                }
                newQueryVecIn.addElement(aWHQuery);
                preparedStatementValuesVecVec.addElement(preparedStatementValuesVec);
             }
          }
       }

       // now exceute all queries
       int totNewQueries = newQueryVecIn.size();
       String sqlString;
       Vector preparedStatementValuesVec;
       for(int i=0; i<totNewQueries; i++) {
          sqlString = (String)newQueryVecIn.elementAt(i);
          log("sqlString="+sqlString);
          preparedStatementValuesVec =
(Vector)preparedStatementValuesVecVec.elementAt(i);
```

```
            int psvvSize = preparedStatementValuesVec.size();
            if(psvvSize > 0) { // this one is to be executed as a PreparedStatement!
               PreparedStatement ps = dbc.createPreparedStatement(sqlString);
               try {
                  for(int j=0; j<psvvSize; j++) {
                     String val = (String)preparedStatementValuesVec.elementAt(j);
                     log("value="+val);
                     ps.setString(j+1, val);
                  }
                  ps.executeQuery();
               }
               catch(Exception e) {
                  log("problem in setting arguments to or executing the prepared
statement, "+ps+":"+e.toString());
                  e.printStackTrace();
                  dbc.doRollback();
                  dbc.setAutoCommitOn();
                  return false;
               }
            }
            else {
               if(dbc.executeUpdate(sqlString) < 1) {
                  dbc.doRollback();
                  dbc.setAutoCommitOn();
                  return false;
               }
            }
         }

         // hardcode
         int modifyDateTime = 1;
         String message = "copy in "+remoteGatewayName;

         String workId = ((DB2Connector)dbc).getWorkId(1);

         // add in the additional WH record to specify that the copy of just prior
inserted records exists in the remote system
         String tableName = tablesOwner + ".WORK_HISTORY";
         sqlString = "INSERT INTO " +tableName+" (WORK_ID, PROBLEM_ID, USER_ID,
WORK_BEGIN_DATE, WORK_BEGIN_TIME, WORK_END_DATE, WORK_END_TIME, DESCRIPTION,
DESC_OVRFLW, MODIFY_DATETIME, TIME_STAMP) SELECT "+workId+", '"+localId+"',
'"+gatewayName+"', date(current timestamp) , time(current timestamp), date(current
timestamp) , time(current timestamp), '"+message+"', ' ', "+modifyDateTime+",
current timestamp FROM "+tablesOwner+".SYSTEM_PROFILE";
         log("sqlString="+sqlString+"\n\n");
         if(dbc.executeUpdate(sqlString) < 1) {
            dbc.doRollback();
            dbc.setAutoCommitOn();
            return false;
         }

         // carry out the insertion into the BRIDGE_CLOSURE table
         tableName = tablesOwner + ".BRIDGE_CLOSURE";
         sqlString = "INSERT INTO " +tableName+" (CLOSURE_ID) VALUES ("+closureId+")";
         log("sqlString="+sqlString+"\n\n");
         if(dbc.executeUpdate(sqlString) < 1) {
            dbc.doRollback();
            dbc.setAutoCommitOn();
            return false;
         }

         // ALL DONE! - commit and set auto commit back to ON
         dbc.commitDatabase();
```

```
      dbc.setAutoCommitOn();

      return true;

   }

   /**
    * this method is implemented by the XMLGateway developer with acknowledgement
code
    * specifically for the target database. It associates the record(ticket) number
in
    * this Gateway's database with the record(ticket) number of the ticket in the
remote database by
    * updating the record(ticket) in this Gateway's database with the
remoteRecordNumber.
    *
    * @param createRecordXML the string(SQL) containing the single quote.
    * @param remoteId the record Id in the remote database.
    * @param localId the record Id in the local database.
    *
    * @return a boolean indicating whether or not the request was successfully
completed.
    */
   public boolean ackRecordInDB(String remoteId, String localId) {
      // unlock the ticket in the database, if an update. Else(create) do this in
the ACKTICKET
      String tableName = tablesOwner + ".PROBLEMS";
      String sqlString = "UPDATE "+tableName+" SET ACTIVE_WITH=null WHERE
PROBLEM_ID='"+localId+"'";
      log("sqlString="+sqlString);
      if(dbc.executeUpdate(sqlString) < 1) {
         log("problem in unlocking update into PROBLEMS.ACTIVE_WITH failed!");
      }
      // unlock the ticket in the database, if an update. Else(create) do this in
the ACKTICKET

      if(vendor_infoTablePresent) {
         tableName = tablesOwner + ".VENDOR_INFO";  // for TSD4.3, etc., that
contain VENDOR_INFO table
      }
      else {
         tableName = tablesOwner + ".PROBLEMS"; // as no VENDOR_INFO table in TSD6.0
but is part of PROBLEMS table
      }

      sqlString = "UPDATE "+tableName+" SET BRIDGE_TICKET_NO='"+remoteId+"' WHERE
PROBLEM_ID='"+localId+"'";
      log("sqlString="+sqlString);
      if(dbc.executeUpdate(sqlString) < 1) {
         dbc.doRollback();
         return false;
      }
      else
         return true;
   }

   public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {

      res.setContentType("text/html");
      res.setHeader("Pragma", "No-cache");
      res.setDateHeader("Expires", 0);
      res.setHeader("Cache-Control", "no-cache");
```

```
      ServletOutputStream out = res.getOutputStream();
      out.println("XML Gateway for TSD loaded.");
      out.close();
   }

   public void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      try {
         BridgeCommunication bc = XMLBridgeCommunicator.parsePost(req, res);
         if(bc.command == BridgeCommunication.CREATETICKET) { // process a create
ticket
            log("creaticket request called");
            String xmldoc = bc.xmldoc;
            String sourceId = bc.sourceID;
            String destId = bc.destID;
            if(xmldoc == null || sourceId == null || destId == null) {
               xmlbc.sendFailure(res, "a problem occurred in creating ticket in the
database due to null values sent in request from bridge.");
               return;
            }

            if(createRecordInDB(xmldoc, sourceId, destId))
               xmlbc.respondCREATETICKET(res);
            else
               xmlbc.sendFailure(res, "a problem occurred in creating a ticket in
the database.") ;
         }
         else if(bc.command == BridgeCommunication.UPDATETICKET) { // process an
update ticket
            log("updateticket request called");
            String xmldoc = bc.xmldoc;
            String sourceId = bc.sourceID;
            String destId = bc.destID;
            if(xmldoc == null || sourceId == null || destId == null) {
               xmlbc.sendFailure(res, "a problem occurred in updating ticket in the
database due to null values sent in request from bridge.");
               return;
            }

            if(updateRecordInDB(xmldoc, sourceId, destId))
               xmlbc.respondUPDATETICKET(res);
            else
               xmlbc.sendFailure(res, "a problem occurred in updating a ticket in
the database or the ticket is currently locked by someone else.") ;
         }
         else if(bc.command == BridgeCommunication.GETTICKETSTOCK) { // process a
get ticket stock
            log("getticketstock request called");
            int count = bc.count;

            if(dbc.getRecordNumberStock(count)) // reserve count number of ticket
numbers in TSD
               xmlbc.respondGETTICKETSTOCK(res, count,
((DB2Connector)dbc).nextTicketNumber);
            else
               xmlbc.sendFailure(res, "a problem occurred in getting ticket stock
from TSD.") ;

         }
         else if(bc.command == BridgeCommunication.ACKTICKET) { // process an ack
ticket
            log("ackticket called");
```

```
            String sourceId = bc.sourceID;
            String destId = bc.destID;
            if(sourceId == null || destId == null) {
               xmlbc.sendFailure(res, "a problem occurred in acking ticket in the
database due to null values sent in request from bridge.");
               return;
            }

            if(ackRecordInDB(destId, sourceId))
               xmlbc.respondACKTICKET(res);
            else
               xmlbc.sendFailure(res, "a problem occurred in ack'ing a ticket in the
database.") ;
         }
         else {
            xmlbc.sendFailure(res, "an unrecognised command sent to the Gateway by
the Bridge.") ;
         }
      }
      catch(Throwable t) {
         log("problem in handling a request from the bridge:"+t.getMessage());
         xmlbc.sendFailure(res, "problem in handling a request from the bridge:
"+t.getMessage()) ;
         t.printStackTrace();
      }
   }

   /**
    * this method can be overridden to do any house cleaning stuff like closing
sockets,
    * database connections, etc.
    *
    * @param msg the output or error message for logging purposes.
    *
    */
   public void destroy() {
      // stop the DB2Connector object
      dbc.alive = false;
      if(dbc!=null) {
         dbc.interrupt();
         dbc.stop();
      }
      // stop the DB2Connector object

      super.destroy();
   }

   public static void main(String argv[]) {
      XMLGWforTSDServlet gw = new XMLGWforTSDServlet();
      try {
         gw.startUpSpecific("e:\\data\\xmlgateway");


         // String xml = "<?xml version=\"1.0\"
encoding=\"UTF-8\"?><TSD_INBOUND_TICKET><GATEWAY><GATEWAY.NEW>1</GATEWAY.NEW><GATEWA
Y.NAME>TSD4.2GatewayA</GATEWAY.NAME><GATEWAY.BRIDGE>LINDYP</GATEWAY.BRIDGE><GATEWAY.
TIMESTAMP>WedMay0316:33:41PDT2000</GATEWAY.TIMESTAMP></GATEWAY><PROBLEM_CLOSURE><PRO
BLEM_CLOSURE.CLOSURE_ID>_</PROBLEM_CLOSURE.CLOSURE_ID><PROBLEM_CLOSURE.TRANSACTIONTY
PE>1</PROBLEM_CLOSURE.TRANSACTIONTYPE><PROBLEM_CLOSURE.CREATECALL>1</PROBLEM_CLOSURE
.CREATECALL><PROBLEM_CLOSURE.CREATEPROBLEM>0</PROBLEM_CLOSURE.CREATEPROBLEM><PROBLEM
_CLOSURE.CLOSEPROBLEM>1</PROBLEM_CLOSURE.CLOSEPROBLEM><PROBLEM_CLOSURE.PICKUPDISPATC
H>0</PROBLEM_CLOSURE.PICKUPDISPATCH><PROBLEM_CLOSURE.TRANSFERUSER>0</PROBLEM_CLOSURE
.TRANSFERUSER><PROBLEM_CLOSURE.DONOTIFICATION>0</PROBLEM_CLOSURE.DONOTIFICATION><PRO
```

BLEM_CLOSURE.EXPLODEGROUP>0</PROBLEM_CLOSURE.EXPLODEGROUP><PROBLEM_CLOSURE.SOLUTIONM
ETHOD>_</PROBLEM_CLOSURE.SOLUTIONMETHOD><PROBLEM_CLOSURE.PROBLEM_ID>00005000</PROBLE
M_CLOSURE.PROBLEM_ID><PROBLEM_CLOSURE.CALL_ID>_</PROBLEM_CLOSURE.CALL_ID><PROBLEM_CL
OSURE.SESSION_ID>_</PROBLEM_CLOSURE.SESSION_ID><PROBLEM_CLOSURE.DISPATCH_ID>_</PROBL
EM_CLOSURE.DISPATCH_ID><PROBLEM_CLOSURE.MODIFY_DATETIME>1</PROBLEM_CLOSURE.MODIFY_DA
TETIME><PROBLEM_CLOSURE.SESSION_BEGIN_DATE>_</PROBLEM_CLOSURE.SESSION_BEGIN_DATE><PR
OBLEM_CLOSURE.SESSION_BEGIN_TIME>_</PROBLEM_CLOSURE.SESSION_BEGIN_TIME><PROBLEM_CLOS
URE.SESSION_END_DATE>_</PROBLEM_CLOSURE.SESSION_END_DATE><PROBLEM_CLOSURE.SESSION_EN
D_TIME>_</PROBLEM_CLOSURE.SESSION_END_TIME><PROBLEM_CLOSURE.LOCATION_ID>_</PROBLEM_C
LOSURE.LOCATION_ID><PROBLEM_CLOSURE.CALLER_ID>_</PROBLEM_CLOSURE.CALLER_ID><PROBLEM_
CLOSURE.CALLER_NAME>fannon</PROBLEM_CLOSURE.CALLER_NAME><PROBLEM_CLOSURE.CALLER_PHON
E>_</PROBLEM_CLOSURE.CALLER_PHONE><PROBLEM_CLOSURE.USER_ID>TSD4.2GatewayA</PROBLEM_C
LOSURE.USER_ID><PROBLEM_CLOSURE.CALL_CODE>LINDYPINCOMING</PROBLEM_CLOSURE.CALL_CODE>
<PROBLEM_CLOSURE.SEVERITY>4</PROBLEM_CLOSURE.SEVERITY><PROBLEM_CLOSURE.PROBLEM_CODE>
TRANSFERRED</PROBLEM_CLOSURE.PROBLEM_CODE><PROBLEM_CLOSURE.PROBLEM_TYPE>LINDYPTSD4.2
Ga</PROBLEM_CLOSURE.PROBLEM_TYPE><PROBLEM_CLOSURE.SYSTEM>HARDWARE</PROBLEM_CLOSURE.S
YSTEM><PROBLEM_CLOSURE.COMPONENT>PersonalSystem</PROBLEM_CLOSURE.COMPONENT><PROBLEM_
CLOSURE.ITEM>processor</PROBLEM_CLOSURE.ITEM><PROBLEM_CLOSURE.MODULE>intel</PROBLEM_
CLOSURE.MODULE><PROBLEM_CLOSURE.DESCRIPTION>Decriptionnnnn</PROBLEM_CLOSURE.DESCRIPT
ION><PROBLEM_CLOSURE.SERIAL_NUMBER>_</PROBLEM_CLOSURE.SERIAL_NUMBER><PROBLEM_CLOSURE
.INVENTORY_ID>_</PROBLEM_CLOSURE.INVENTORY_ID><PROBLEM_CLOSURE.PROBLEM_RESULT>_</PRO
BLEM_CLOSURE.PROBLEM_RESULT><PROBLEM_CLOSURE.TIME_SPENT>_</PROBLEM_CLOSURE.TIME_SPEN
T><PROBLEM_CLOSURE.DIAG_NODE>_</PROBLEM_CLOSURE.DIAG_NODE><PROBLEM_CLOSURE.FLX_CAL_V
CHR1>_</PROBLEM_CLOSURE.FLX_CAL_VCHR1><PROBLEM_CLOSURE.FLX_CAL_VCHR2>_</PROBLEM_CLOS
URE.FLX_CAL_VCHR2><PROBLEM_CLOSURE.FLX_CAL_VCHR3>_</PROBLEM_CLOSURE.FLX_CAL_VCHR3><P
ROBLEM_CLOSURE.FLX_CAL_VCHR4>_</PROBLEM_CLOSURE.FLX_CAL_VCHR4><PROBLEM_CLOSURE.FLX_C
AL_INT1>_</PROBLEM_CLOSURE.FLX_CAL_INT1><PROBLEM_CLOSURE.FLX_CAL_INT2>_</PROBLEM_CLO
SURE.FLX_CAL_INT2><PROBLEM_CLOSURE.FLX_CAL_INT3>_</PROBLEM_CLOSURE.FLX_CAL_INT3><PRO
BLEM_CLOSURE.FLX_CAL_INT4>_</PROBLEM_CLOSURE.FLX_CAL_INT4><PROBLEM_CLOSURE.FLX_CAL_D
ATE1>_</PROBLEM_CLOSURE.FLX_CAL_DATE1><PROBLEM_CLOSURE.FLX_CAL_DATE2>_</PROBLEM_CLOS
URE.FLX_CAL_DATE2><PROBLEM_CLOSURE.FLX_CAL_TIME1>_</PROBLEM_CLOSURE.FLX_CAL_TIME1><P
ROBLEM_CLOSURE.FLX_CAL_TIME2>_</PROBLEM_CLOSURE.FLX_CAL_TIME2><PROBLEM_CLOSURE.FLX_P
RO_VCHR1>_</PROBLEM_CLOSURE.FLX_PRO_VCHR1><PROBLEM_CLOSURE.FLX_PRO_VCHR2>_</PROBLEM_
CLOSURE.FLX_PRO_VCHR2><PROBLEM_CLOSURE.FLX_PRO_VCHR3>_</PROBLEM_CLOSURE.FLX_PRO_VCHR
3><PROBLEM_CLOSURE.FLX_PRO_VCHR4>_</PROBLEM_CLOSURE.FLX_PRO_VCHR4><PROBLEM_CLOSURE.F
LX_PRO_INT1>_</PROBLEM_CLOSURE.FLX_PRO_INT1><PROBLEM_CLOSURE.FLX_PRO_INT2>_</PROBLEM
_CLOSURE.FLX_PRO_INT2><PROBLEM_CLOSURE.FLX_PRO_INT3>_</PROBLEM_CLOSURE.FLX_PRO_INT3>
<PROBLEM_CLOSURE.FLX_PRO_INT4>_</PROBLEM_CLOSURE.FLX_PRO_INT4><PROBLEM_CLOSURE.FLX_P
RO_DATE1>_</PROBLEM_CLOSURE.FLX_PRO_DATE1><PROBLEM_CLOSURE.FLX_PRO_DATE2>_</PROBLEM_
CLOSURE.FLX_PRO_DATE2><PROBLEM_CLOSURE.FLX_PRO_TIME1>_</PROBLEM_CLOSURE.FLX_PRO_TIME
1><PROBLEM_CLOSURE.FLX_PRO_TIME2>_</PROBLEM_CLOSURE.FLX_PRO_TIME2><PROBLEM_CLOSURE.R
CV_GROUP_ID>_</PROBLEM_CLOSURE.RCV_GROUP_ID><PROBLEM_CLOSURE.RCV_USER_ID>_</PROBLEM_
CLOSURE.RCV_USER_ID><PROBLEM_CLOSURE.LINE_NUMBER>_</PROBLEM_CLOSURE.LINE_NUMBER><PRO
BLEM_CLOSURE.NOTIFICATION_DATE>_</PROBLEM_CLOSURE.NOTIFICATION_DATE><PROBLEM_CLOSURE
.NOTIFICATION_TIME>_</PROBLEM_CLOSURE.NOTIFICATION_TIME><PROBLEM_CLOSURE.NOTIFICATIO
N_TYPE>_</PROBLEM_CLOSURE.NOTIFICATION_TYPE><PROBLEM_CLOSURE.NOTIFY_CONTACT>_</PROBL
EM_CLOSURE.NOTIFY_CONTACT><PROBLEM_CLOSURE.SOLUTION>_</PROBLEM_CLOSURE.SOLUTION><PRO
BLEM_CLOSURE.ACTIVE>0</PROBLEM_CLOSURE.ACTIVE><PROBLEM_CLOSURE.SOLUTION_ID>_</PROBLE
M_CLOSURE.SOLUTION_ID><PROBLEM_CLOSURE.AID_TYPE>_</PROBLEM_CLOSURE.AID_TYPE><PROBLEM
_CLOSURE.ANNOTATION_FILE>_</PROBLEM_CLOSURE.ANNOTATION_FILE><PROBLEM_CLOSURE.CONTROL
TIME></PROBLEM_CLOSURE.CONTROL_TIME><PROBLEM_CLOSURE.FLX_PRO_VCHR5>_</PROBLEM_CLOS
URE.FLX_PRO_VCHR5><PROBLEM_CLOSURE.GROUP_ID>LINDYP</PROBLEM_CLOSURE.GROUP_ID><PROBLE
M_CLOSURE.LOGGED_USER>TSD4.2GatewayA</PROBLEM_CLOSURE.LOGGED_USER></PROBLEM_CLOSURE>
<PROBLEM_CLOSURE2><PROBLEM_CLOSURE2.CLOSURE_ID>_</PROBLEM_CLOSURE2.CLOSURE_ID><PROBL
EM_CLOSURE2.DOCUMENT_ID>_</PROBLEM_CLOSURE2.DOCUMENT_ID><PROBLEM_CLOSURE2.VEND_FIELD
1>_</PROBLEM_CLOSURE2.VEND_FIELD1><PROBLEM_CLOSURE2.VEND_FIELD2>_</PROBLEM_CLOSURE2.
VEND_FIELD2><PROBLEM_CLOSURE2.VEND_FIELD3>_</PROBLEM_CLOSURE2.VEND_FIELD3><PROBLEM_C
LOSURE2.VEND_FIELD4>_</PROBLEM_CLOSURE2.VEND_FIELD4><PROBLEM_CLOSURE2.VEND_FIELD5>_<
/PROBLEM_CLOSURE2.VEND_FIELD5><PROBLEM_CLOSURE2.VEND_FIELD6>_</PROBLEM_CLOSURE2.VEND
_FIELD6><PROBLEM_CLOSURE2.VEND_FIELD7>_</PROBLEM_CLOSURE2.VEND_FIELD7><PROBLEM_CLOSU
RE2.VEND_FIELD8>_</PROBLEM_CLOSURE2.VEND_FIELD8><PROBLEM_CLOSURE2.VEND_FIELD9>_</PRO
BLEM_CLOSURE2.VEND_FIELD9><PROBLEM_CLOSURE2.VEND_FIELD10>_</PROBLEM_CLOSURE2.VEND_FI
ELD10><PROBLEM_CLOSURE2.DATETIME1>_</PROBLEM_CLOSURE2.DATETIME1><PROBLEM_CLOSURE2.DA

TETIME2>_</PROBLEM_CLOSURE2.DATETIME2><PROBLEM_CLOSURE2.DATETIME3>_</PROBLEM_CLOSURE
2.DATETIME3><PROBLEM_CLOSURE2.DATETIME4>_</PROBLEM_CLOSURE2.DATETIME4><PROBLEM_CLOSU
RE2.DATETIME5>_</PROBLEM_CLOSURE2.DATETIME5><PROBLEM_CLOSURE2.DATETIME6>_</PROBLEM_C
LOSURE2.DATETIME6><PROBLEM_CLOSURE2.DATETIME7>_</PROBLEM_CLOSURE2.DATETIME7><PROBLEM
CLOSURE2.DATETIME8></PROBLEM_CLOSURE2.DATETIME8><PROBLEM_CLOSURE2.DATETIME9>_</PRO
BLEM_CLOSURE2.DATETIME9><PROBLEM_CLOSURE2.DATETIME10>_</PROBLEM_CLOSURE2.DATETIME10>
<PROBLEM_CLOSURE2.BRIDGE_ID>LINDYP</PROBLEM_CLOSURE2.BRIDGE_ID><PROBLEM_CLOSURE2.BRI
DGE_TICKET_NO>00002864</PROBLEM_CLOSURE2.BRIDGE_TICKET_NO><PROBLEM_CLOSURE2.INCOMING
_FLAG>0</PROBLEM_CLOSURE2.INCOMING_FLAG><PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION>LINDYP<
/PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION><PROBLEM_CLOSURE2.CUSTOMER_ID>_</PROBLEM_CLOSUR
E2.CUSTOMER_ID><PROBLEM_CLOSURE2.ACCOUNT_ID>_</PROBLEM_CLOSURE2.ACCOUNT_ID><PROBLEM_
CLOSURE2.REPORTER_ID>_</PROBLEM_CLOSURE2.REPORTER_ID><PROBLEM_CLOSURE2.RESOLVER_ID>_
</PROBLEM_CLOSURE2.RESOLVER_ID><PROBLEM_CLOSURE2.REPORTER_GROUP>_</PROBLEM_CLOSURE2.
REPORTER_GROUP><PROBLEM_CLOSURE2.RESOLVER_GROUP>_</PROBLEM_CLOSURE2.RESOLVER_GROUP><
PROBLEM_CLOSURE2.TEAM>_</PROBLEM_CLOSURE2.TEAM><PROBLEM_CLOSURE2.CALLBACK_DATE>_</PR
OBLEM_CLOSURE2.CALLBACK_DATE><PROBLEM_CLOSURE2.CALLBACK_TIME>_</PROBLEM_CLOSURE2.CAL
LBACK_TIME><PROBLEM_CLOSURE2.ORIG_TARGET_DATE>_</PROBLEM_CLOSURE2.ORIG_TARGET_DATE><
PROBLEM_CLOSURE2.ORIG_TARGET_TIME>_</PROBLEM_CLOSURE2.ORIG_TARGET_TIME><PROBLEM_CLOS
URE2.CURR_TARGET_DATE>_</PROBLEM_CLOSURE2.CURR_TARGET_DATE><PROBLEM_CLOSURE2.CURR_TA
RGET_TIME>_</PROBLEM_CLOSURE2.CURR_TARGET_TIME><PROBLEM_CLOSURE2.OCCURRED_DATE>2000-
05-03</PROBLEM_CLOSURE2.OCCURRED_DATE><PROBLEM_CLOSURE2.OCCURRED_TIME>23:33:52</PROB
LEM_CLOSURE2.OCCURRED_TIME><PROBLEM_CLOSURE2.SOLVED_DATE>_</PROBLEM_CLOSURE2.SOLVED_
DATE><PROBLEM_CLOSURE2.SOLVED_TIME>_</PROBLEM_CLOSURE2.SOLVED_TIME><PROBLEM_CLOSURE2
.CAUSE_CHANGE_NO>_</PROBLEM_CLOSURE2.CAUSE_CHANGE_NO><PROBLEM_CLOSURE2.ORIGINAL_SEVE
RITY>4</PROBLEM_CLOSURE2.ORIGINAL_SEVERITY><PROBLEM_CLOSURE2.DURATION>_</PROBLEM_CLO
SURE2.DURATION><PROBLEM_CLOSURE2.NODEID>_</PROBLEM_CLOSURE2.NODEID><PROBLEM_CLOSURE2
.REASSIGNMENT>_</PROBLEM_CLOSURE2.REASSIGNMENT><PROBLEM_CLOSURE2.PROBLEM_ABSTRACT>De
criptionnnnn</PROBLEM_CLOSURE2.PROBLEM_ABSTRACT><PROBLEM_CLOSURE2.PROBLEM_DUPLICATE>
_</PROBLEM_CLOSURE2.PROBLEM_DUPLICATE><PROBLEM_CLOSURE2.RCA_REQUIRED>_</PROBLEM_CLOS
URE2.RCA_REQUIRED><PROBLEM_CLOSURE2.PREV_OCCURRED_DATE>_</PROBLEM_CLOSURE2.PREV_OCCU
RRED_DATE><PROBLEM_CLOSURE2.PREV_OCCURRED_TIME>_</PROBLEM_CLOSURE2.PREV_OCCURRED_TIM
E><PROBLEM_CLOSURE2.PREV_CT_DATE>_</PROBLEM_CLOSURE2.PREV_CT_DATE><PROBLEM_CLOSURE2.
PREV_CT_TIME>_</PROBLEM_CLOSURE2.PREV_CT_TIME><PROBLEM_CLOSURE2.CAUSE_CODE>_</PROBLE
M_CLOSURE2.CAUSE_CODE><PROBLEM_CLOSURE2.RCA_TEXT>_</PROBLEM_CLOSURE2.RCA_TEXT><PROBL
EM_CLOSURE2.REASON>_</PROBLEM_CLOSURE2.REASON></PROBLEM_CLOSURE2><HISTORY><WORK_HIST
ORY.WORK_ID>_</WORK_HISTORY.WORK_ID><WORK_HISTORY.PROBLEM_ID>00005000</WORK_HISTORY.
PROBLEM_ID><WORK_HISTORY.USER_ID>TSD4.2GatewayA</WORK_HISTORY.USER_ID><WORK_HISTORY.
WORK_BEGIN_DATE>2000-05-03</WORK_HISTORY.WORK_BEGIN_DATE><WORK_HISTORY.WORK_BEGIN_TI
ME>23:34:39</WORK_HISTORY.WORK_BEGIN_TIME><WORK_HISTORY.WORK_END_DATE>2000-05-03</WO
RK_HISTORY.WORK_END_DATE><WORK_HISTORY.WORK_END_TIME>23:34:39</WORK_HISTORY.WORK_END
_TIME><WORK_HISTORY.DESCRIPTION>hello</WORK_HISTORY.DESCRIPTION><WORK_HISTORY.DESC_O
VRFLW>_</WORK_HISTORY.DESC_OVRFLW><WORK_HISTORY.MODIFY_DATETIME>1</WORK_HISTORY.MODI
FY_DATETIME><WORK_HISTORY.TIME_STAMP>2000-05-0319:34:52.351299</WORK_HISTORY.TIME_ST
AMP><WORK_HISTORY.ACTIVITY_ACTION_ID>_</WORK_HISTORY.ACTIVITY_ACTION_ID></HISTORY><H
ISTORY><WORK_HISTORY.WORK_ID>_</WORK_HISTORY.WORK_ID><WORK_HISTORY.PROBLEM_ID>000050
00</WORK_HISTORY.PROBLEM_ID><WORK_HISTORY.USER_ID>TSD4.2GatewayA</WORK_HISTORY.USER_
ID><WORK_HISTORY.WORK_BEGIN_DATE>2000-05-03</WORK_HISTORY.WORK_BEGIN_DATE><WORK_HIST
ORY.WORK_BEGIN_TIME>23:34:54</WORK_HISTORY.WORK_BEGIN_TIME><WORK_HISTORY.WORK_END_DA
TE>2000-05-03</WORK_HISTORY.WORK_END_DATE><WORK_HISTORY.WORK_END_TIME>23:34:54</WORK
_HISTORY.WORK_END_TIME><WORK_HISTORY.DESCRIPTION>there</WORK_HISTORY.DESCRIPTION><WO
RK_HISTORY.DESC_OVRFLW>_</WORK_HISTORY.DESC_OVRFLW><WORK_HISTORY.MODIFY_DATETIME>1</
WORK_HISTORY.MODIFY_DATETIME><WORK_HISTORY.TIME_STAMP>2000-05-0319:35:02.504746</WOR
K_HISTORY.TIME_STAMP><WORK_HISTORY.ACTIVITY_ACTION_ID>_</WORK_HISTORY.ACTIVITY_ACTIO
N_ID></HISTORY><HISTORY><WORK_HISTORY.WORK_ID>_</WORK_HISTORY.WORK_ID><WORK_HISTORY.
PROBLEM_ID>00005000</WORK_HISTORY.PROBLEM_ID><WORK_HISTORY.USER_ID>TSD4.2GatewayA</W
ORK_HISTORY.USER_ID><WORK_HISTORY.WORK_BEGIN_DATE>2000-05-03</WORK_HISTORY.WORK_BEGI
N_DATE><WORK_HISTORY.WORK_BEGIN_TIME>23:33:52</WORK_HISTORY.WORK_BEGIN_TIME><WORK_HI
STORY.WORK_END_DATE>2000-05-03</WORK_HISTORY.WORK_END_DATE><WORK_HISTORY.WORK_END_TI
ME>23:33:52</WORK_HISTORY.WORK_END_TIME><WORK_HISTORY.DESCRIPTION>TSD4.2GatewayA:ACT
IVE_SLA_.ACTIVE_SLA_ID=_:ACTIVE_SLA.TERM_ID=_:ACTIVE_SLA.REFERENCE_ID=_:ACTIVE_SLA.R
ELATIVE_TO=_:ACTIVE_SLA.SLA_STATE=_:ACTIVE_SLA.BREACH_DATE=_:ACTIVE_SLA.BREACH_TIME=
_:ACTIVE_SLA.NEXT_SCHEDULE_ID=_:ACTIVE_SLA.NEXT_FIRE_DATE=_:ACTIVE_SLA.NEXT_FIRE_TIM
E=_</WORK_HISTORY.DESCRIPTION><WORK_HISTORY.DESC_OVRFLW>_</WORK_HISTORY.DESC_OVRFLW>

```
<WORK_HISTORY.MODIFY_DATETIME>1</WORK_HISTORY.MODIFY_DATETIME><WORK_HISTORY.TIME_STA
MP>_</WORK_HISTORY.TIME_STAMP><WORK_HISTORY.ACTIVITY_ACTION_ID>_</WORK_HISTORY.ACTIV
ITY_ACTION_ID></HISTORY></TSD_INBOUND_TICKET>";

        String xml = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<TSD_INBOUND_TICKET> <GATEWAY> <GATEWAY.NEW> 1 </GATEWAY.NEW> <GATEWAY.NAME>
TSD4.2GatewayA </GATEWAY.NAME> <GATEWAY.BRIDGE> LINDYP </GATEWAY.BRIDGE>
<GATEWAY.TIMESTAMP> Wed May 03 16:33:41 PDT 2000 </GATEWAY.TIMESTAMP> </GATEWAY>
<PROBLEM_CLOSURE> <PROBLEM_CLOSURE.CLOSURE_ID> _ </PROBLEM_CLOSURE.CLOSURE_ID>
<PROBLEM_CLOSURE.TRANSACTIONTYPE> 1 </PROBLEM_CLOSURE.TRANSACTIONTYPE>
<PROBLEM_CLOSURE.CREATECALL> 1 </PROBLEM_CLOSURE.CREATECALL>
<PROBLEM_CLOSURE.CREATEPROBLEM> 0 </PROBLEM_CLOSURE.CREATEPROBLEM>
<PROBLEM_CLOSURE.CLOSEPROBLEM> 1 </PROBLEM_CLOSURE.CLOSEPROBLEM>
<PROBLEM_CLOSURE.PICKUPDISPATCH> 0 </PROBLEM_CLOSURE.PICKUPDISPATCH>
<PROBLEM_CLOSURE.TRANSFERUSER> 0 </PROBLEM_CLOSURE.TRANSFERUSER>
<PROBLEM_CLOSURE.DONOTIFICATION> 0 </PROBLEM_CLOSURE.DONOTIFICATION>
<PROBLEM_CLOSURE.EXPLODEGROUP> 0 </PROBLEM_CLOSURE.EXPLODEGROUP>
<PROBLEM_CLOSURE.SOLUTIONMETHOD> _ </PROBLEM_CLOSURE.SOLUTIONMETHOD>
<PROBLEM_CLOSURE.PROBLEM_ID> 00005000 </PROBLEM_CLOSURE.PROBLEM_ID>
<PROBLEM_CLOSURE.CALL_ID> _ </PROBLEM_CLOSURE.CALL_ID> <PROBLEM_CLOSURE.SESSION_ID>
_ </PROBLEM_CLOSURE.SESSION_ID> <PROBLEM_CLOSURE.DISPATCH_ID> _
</PROBLEM_CLOSURE.DISPATCH_ID> <PROBLEM_CLOSURE.MODIFY_DATETIME> 1
</PROBLEM_CLOSURE.MODIFY_DATETIME> <PROBLEM_CLOSURE.SESSION_BEGIN_DATE> _
</PROBLEM_CLOSURE.SESSION_BEGIN_DATE> <PROBLEM_CLOSURE.SESSION_BEGIN_TIME> _
</PROBLEM_CLOSURE.SESSION_BEGIN_TIME> <PROBLEM_CLOSURE.SESSION_END_DATE> _
</PROBLEM_CLOSURE.SESSION_END_DATE> <PROBLEM_CLOSURE.SESSION_END_TIME> _
</PROBLEM_CLOSURE.SESSION_END_TIME> <PROBLEM_CLOSURE.LOCATION_ID> _
</PROBLEM_CLOSURE.LOCATION_ID> <PROBLEM_CLOSURE.CALLER_ID> _
</PROBLEM_CLOSURE.CALLER_ID> <PROBLEM_CLOSURE.CALLER_NAME> fannon
</PROBLEM_CLOSURE.CALLER_NAME> <PROBLEM_CLOSURE.CALLER_PHONE> _
</PROBLEM_CLOSURE.CALLER_PHONE> <PROBLEM_CLOSURE.USER_ID> TSD4.2GatewayA
</PROBLEM_CLOSURE.USER_ID> <PROBLEM_CLOSURE.CALL_CODE> LINDYPINCOMING
</PROBLEM_CLOSURE.CALL_CODE> <PROBLEM_CLOSURE.SEVERITY> 4
</PROBLEM_CLOSURE.SEVERITY> <PROBLEM_CLOSURE.PROBLEM_CODE> TRANSFERRED
</PROBLEM_CLOSURE.PROBLEM_CODE> <PROBLEM_CLOSURE.PROBLEM_TYPE> LINDYP TSD4.2Ga
</PROBLEM_CLOSURE.PROBLEM_TYPE> <PROBLEM_CLOSURE.SYSTEM> HARDWARE
</PROBLEM_CLOSURE.SYSTEM> <PROBLEM_CLOSURE.COMPONENT> Personal System
</PROBLEM_CLOSURE.COMPONENT> <PROBLEM_CLOSURE.ITEM> processor
</PROBLEM_CLOSURE.ITEM> <PROBLEM_CLOSURE.MODULE> intel </PROBLEM_CLOSURE.MODULE>
<PROBLEM_CLOSURE.DESCRIPTION> Decriptionnnnn </PROBLEM_CLOSURE.DESCRIPTION>
<PROBLEM_CLOSURE.SERIAL_NUMBER> _ </PROBLEM_CLOSURE.SERIAL_NUMBER>
<PROBLEM_CLOSURE.INVENTORY_ID> _ </PROBLEM_CLOSURE.INVENTORY_ID>
<PROBLEM_CLOSURE.PROBLEM_RESULT> _ </PROBLEM_CLOSURE.PROBLEM_RESULT>
<PROBLEM_CLOSURE.TIME_SPENT> _ </PROBLEM_CLOSURE.TIME_SPENT>
<PROBLEM_CLOSURE.DIAG_NODE> _ </PROBLEM_CLOSURE.DIAG_NODE>
<PROBLEM_CLOSURE.FLX_CAL_VCHR1> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR1>
<PROBLEM_CLOSURE.FLX_CAL_VCHR2> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR2>
<PROBLEM_CLOSURE.FLX_CAL_VCHR3> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR3>
<PROBLEM_CLOSURE.FLX_CAL_VCHR4> _ </PROBLEM_CLOSURE.FLX_CAL_VCHR4>
<PROBLEM_CLOSURE.FLX_CAL_INT1> _ </PROBLEM_CLOSURE.FLX_CAL_INT1>
<PROBLEM_CLOSURE.FLX_CAL_INT2> _ </PROBLEM_CLOSURE.FLX_CAL_INT2>
<PROBLEM_CLOSURE.FLX_CAL_INT3> _ </PROBLEM_CLOSURE.FLX_CAL_INT3>
<PROBLEM_CLOSURE.FLX_CAL_INT4> _ </PROBLEM_CLOSURE.FLX_CAL_INT4>
<PROBLEM_CLOSURE.FLX_CAL_DATE1> _ </PROBLEM_CLOSURE.FLX_CAL_DATE1>
<PROBLEM_CLOSURE.FLX_CAL_DATE2> _ </PROBLEM_CLOSURE.FLX_CAL_DATE2>
<PROBLEM_CLOSURE.FLX_CAL_TIME1> _ </PROBLEM_CLOSURE.FLX_CAL_TIME1>
<PROBLEM_CLOSURE.FLX_CAL_TIME2> _ </PROBLEM_CLOSURE.FLX_CAL_TIME2>
<PROBLEM_CLOSURE.FLX_PRO_VCHR1> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR1>
<PROBLEM_CLOSURE.FLX_PRO_VCHR2> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR2>
<PROBLEM_CLOSURE.FLX_PRO_VCHR3> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR3>
<PROBLEM_CLOSURE.FLX_PRO_VCHR4> _ </PROBLEM_CLOSURE.FLX_PRO_VCHR4>
<PROBLEM_CLOSURE.FLX_PRO_INT1> _ </PROBLEM_CLOSURE.FLX_PRO_INT1>
<PROBLEM_CLOSURE.FLX_PRO_INT2> _ </PROBLEM_CLOSURE.FLX_PRO_INT2>
```

```
<PROBLEM_CLOSURE.FLX_PRO_INT3> _ </PROBLEM_CLOSURE.FLX_PRO_INT3>
<PROBLEM_CLOSURE.FLX_PRO_INT4> _ </PROBLEM_CLOSURE.FLX_PRO_INT4>
<PROBLEM_CLOSURE.FLX_PRO_DATE1> _ </PROBLEM_CLOSURE.FLX_PRO_DATE1>
<PROBLEM_CLOSURE.FLX_PRO_DATE2> _ </PROBLEM_CLOSURE.FLX_PRO_DATE2>
<PROBLEM_CLOSURE.FLX_PRO_TIME1> _ </PROBLEM_CLOSURE.FLX_PRO_TIME1>
<PROBLEM_CLOSURE.FLX_PRO_TIME2> _ </PROBLEM_CLOSURE.FLX_PRO_TIME2>
<PROBLEM_CLOSURE.RCV_GROUP_ID> _ </PROBLEM_CLOSURE.RCV_GROUP_ID>
<PROBLEM_CLOSURE.RCV_USER_ID> _ </PROBLEM_CLOSURE.RCV_USER_ID>
<PROBLEM_CLOSURE.LINE_NUMBER> _ </PROBLEM_CLOSURE.LINE_NUMBER>
<PROBLEM_CLOSURE.NOTIFICATION_DATE> _ </PROBLEM_CLOSURE.NOTIFICATION_DATE>
<PROBLEM_CLOSURE.NOTIFICATION_TIME> _ </PROBLEM_CLOSURE.NOTIFICATION_TIME>
<PROBLEM_CLOSURE.NOTIFICATION_TYPE> _ </PROBLEM_CLOSURE.NOTIFICATION_TYPE>
<PROBLEM_CLOSURE.NOTIFY_CONTACT> _ </PROBLEM_CLOSURE.NOTIFY_CONTACT>
<PROBLEM_CLOSURE.SOLUTION> _ </PROBLEM_CLOSURE.SOLUTION> <PROBLEM_CLOSURE.ACTIVE> 0
</PROBLEM_CLOSURE.ACTIVE> <PROBLEM_CLOSURE.SOLUTION_ID> _
</PROBLEM_CLOSURE.SOLUTION_ID> <PROBLEM_CLOSURE.AID_TYPE> _
</PROBLEM_CLOSURE.AID_TYPE> <PROBLEM_CLOSURE.ANNOTATION_FILE> _
</PROBLEM_CLOSURE.ANNOTATION_FILE> <PROBLEM_CLOSURE.CONTROL_TIME> _
</PROBLEM_CLOSURE.CONTROL_TIME> <PROBLEM_CLOSURE.FLX_PRO_VCHR5> _
</PROBLEM_CLOSURE.FLX_PRO_VCHR5> <PROBLEM_CLOSURE.GROUP_ID> LINDYP
</PROBLEM_CLOSURE.GROUP_ID> <PROBLEM_CLOSURE.LOGGED_USER> TSD4.2GatewayA
</PROBLEM_CLOSURE.LOGGED_USER> </PROBLEM_CLOSURE> <PROBLEM_CLOSURE2>
<PROBLEM_CLOSURE2.CLOSURE_ID> _ </PROBLEM_CLOSURE2.CLOSURE_ID>
<PROBLEM_CLOSURE2.DOCUMENT_ID> _ </PROBLEM_CLOSURE2.DOCUMENT_ID>
<PROBLEM_CLOSURE2.VEND_FIELD1> _ </PROBLEM_CLOSURE2.VEND_FIELD1>
<PROBLEM_CLOSURE2.VEND_FIELD2> _ </PROBLEM_CLOSURE2.VEND_FIELD2>
<PROBLEM_CLOSURE2.VEND_FIELD3> _ </PROBLEM_CLOSURE2.VEND_FIELD3>
<PROBLEM_CLOSURE2.VEND_FIELD4> _ </PROBLEM_CLOSURE2.VEND_FIELD4>
<PROBLEM_CLOSURE2.VEND_FIELD5> _ </PROBLEM_CLOSURE2.VEND_FIELD5>
<PROBLEM_CLOSURE2.VEND_FIELD6> _ </PROBLEM_CLOSURE2.VEND_FIELD6>
<PROBLEM_CLOSURE2.VEND_FIELD7> _ </PROBLEM_CLOSURE2.VEND_FIELD7>
<PROBLEM_CLOSURE2.VEND_FIELD8> _ </PROBLEM_CLOSURE2.VEND_FIELD8>
<PROBLEM_CLOSURE2.VEND_FIELD9> _ </PROBLEM_CLOSURE2.VEND_FIELD9>
<PROBLEM_CLOSURE2.VEND_FIELD10> _ </PROBLEM_CLOSURE2.VEND_FIELD10>
<PROBLEM_CLOSURE2.DATETIME1> _ </PROBLEM_CLOSURE2.DATETIME1>
<PROBLEM_CLOSURE2.DATETIME2> _ </PROBLEM_CLOSURE2.DATETIME2>
<PROBLEM_CLOSURE2.DATETIME3> _ </PROBLEM_CLOSURE2.DATETIME3>
<PROBLEM_CLOSURE2.DATETIME4> _ </PROBLEM_CLOSURE2.DATETIME4>
<PROBLEM_CLOSURE2.DATETIME5> _ </PROBLEM_CLOSURE2.DATETIME5>
<PROBLEM_CLOSURE2.DATETIME6> _ </PROBLEM_CLOSURE2.DATETIME6>
<PROBLEM_CLOSURE2.DATETIME7> _ </PROBLEM_CLOSURE2.DATETIME7>
<PROBLEM_CLOSURE2.DATETIME8> _ </PROBLEM_CLOSURE2.DATETIME8>
<PROBLEM_CLOSURE2.DATETIME9> _ </PROBLEM_CLOSURE2.DATETIME9>
<PROBLEM_CLOSURE2.DATETIME10> _ </PROBLEM_CLOSURE2.DATETIME10>
<PROBLEM_CLOSURE2.BRIDGE_ID> LINDYP </PROBLEM_CLOSURE2.BRIDGE_ID>
<PROBLEM_CLOSURE2.BRIDGE_TICKET_NO> 00002864 </PROBLEM_CLOSURE2.BRIDGE_TICKET_NO>
<PROBLEM_CLOSURE2.INCOMING_FLAG> 0 </PROBLEM_CLOSURE2.INCOMING_FLAG>
<PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION> LINDYP </PROBLEM_CLOSURE2.BRIDGE_DESCRIPTION>
<PROBLEM_CLOSURE2.CUSTOMER_ID> _ </PROBLEM_CLOSURE2.CUSTOMER_ID>
<PROBLEM_CLOSURE2.ACCOUNT_ID> _ </PROBLEM_CLOSURE2.ACCOUNT_ID>
<PROBLEM_CLOSURE2.REPORTER_ID> _ </PROBLEM_CLOSURE2.REPORTER_ID>
<PROBLEM_CLOSURE2.RESOLVER_ID> _ </PROBLEM_CLOSURE2.RESOLVER_ID>
<PROBLEM_CLOSURE2.REPORTER_GROUP> _ </PROBLEM_CLOSURE2.REPORTER_GROUP>
<PROBLEM_CLOSURE2.RESOLVER_GROUP> _ </PROBLEM_CLOSURE2.RESOLVER_GROUP>
<PROBLEM_CLOSURE2.TEAM> _ </PROBLEM_CLOSURE2.TEAM> <PROBLEM_CLOSURE2.CALLBACK_DATE>
_ </PROBLEM_CLOSURE2.CALLBACK_DATE> <PROBLEM_CLOSURE2.CALLBACK_TIME> _
</PROBLEM_CLOSURE2.CALLBACK_TIME> <PROBLEM_CLOSURE2.ORIG_TARGET_DATE> _
</PROBLEM_CLOSURE2.ORIG_TARGET_DATE> <PROBLEM_CLOSURE2.ORIG_TARGET_TIME> _
</PROBLEM_CLOSURE2.ORIG_TARGET_TIME> <PROBLEM_CLOSURE2.CURR_TARGET_DATE> _
</PROBLEM_CLOSURE2.CURR_TARGET_DATE> <PROBLEM_CLOSURE2.CURR_TARGET_TIME> _
</PROBLEM_CLOSURE2.CURR_TARGET_TIME> <PROBLEM_CLOSURE2.OCCURRED_DATE> 2000-05-03
</PROBLEM_CLOSURE2.OCCURRED_DATE> <PROBLEM_CLOSURE2.OCCURRED_TIME> 23:33:52
</PROBLEM_CLOSURE2.OCCURRED_TIME> <PROBLEM_CLOSURE2.SOLVED_DATE> _
```

```
</PROBLEM_CLOSURE2.SOLVED_DATE> <PROBLEM_CLOSURE2.SOLVED_TIME> _
</PROBLEM_CLOSURE2.SOLVED_TIME> <PROBLEM_CLOSURE2.CAUSE_CHANGE_NO> _
</PROBLEM_CLOSURE2.CAUSE_CHANGE_NO> <PROBLEM_CLOSURE2.ORIGINAL_SEVERITY> 4
</PROBLEM_CLOSURE2.ORIGINAL_SEVERITY> <PROBLEM_CLOSURE2.DURATION> _
</PROBLEM_CLOSURE2.DURATION> <PROBLEM_CLOSURE2.NODEID> _ </PROBLEM_CLOSURE2.NODEID>
<PROBLEM_CLOSURE2.REASSIGNMENT> _ </PROBLEM_CLOSURE2.REASSIGNMENT>
<PROBLEM_CLOSURE2.PROBLEM_ABSTRACT> Decriptionnnnn
</PROBLEM_CLOSURE2.PROBLEM_ABSTRACT> <PROBLEM_CLOSURE2.PROBLEM_DUPLICATE> _
</PROBLEM_CLOSURE2.PROBLEM_DUPLICATE> <PROBLEM_CLOSURE2.RCA_REQUIRED> _
</PROBLEM_CLOSURE2.RCA_REQUIRED> <PROBLEM_CLOSURE2.PREV_OCCURRED_DATE> _
</PROBLEM_CLOSURE2.PREV_OCCURRED_DATE> <PROBLEM_CLOSURE2.PREV_OCCURRED_TIME> _
</PROBLEM_CLOSURE2.PREV_OCCURRED_TIME> <PROBLEM_CLOSURE2.PREV_CT_DATE> _
</PROBLEM_CLOSURE2.PREV_CT_DATE> <PROBLEM_CLOSURE2.PREV_CT_TIME> _
</PROBLEM_CLOSURE2.PREV_CT_TIME> <PROBLEM_CLOSURE2.CAUSE_CODE> _
</PROBLEM_CLOSURE2.CAUSE_CODE> <PROBLEM_CLOSURE2.RCA_TEXT> _
</PROBLEM_CLOSURE2.RCA_TEXT> <PROBLEM_CLOSURE2.REASON> _ </PROBLEM_CLOSURE2.REASON>
</PROBLEM_CLOSURE2> <HISTORY> <WORK_HISTORY.WORK_ID> _ </WORK_HISTORY.WORK_ID>
<WORK_HISTORY.PROBLEM_ID> 00005000 </WORK_HISTORY.PROBLEM_ID> <WORK_HISTORY.USER_ID>
TSD4.2GatewayA </WORK_HISTORY.USER_ID> <WORK_HISTORY.WORK_BEGIN_DATE> 2000-05-03
</WORK_HISTORY.WORK_BEGIN_DATE> <WORK_HISTORY.WORK_BEGIN_TIME> 23:34:39
</WORK_HISTORY.WORK_BEGIN_TIME> <WORK_HISTORY.WORK_END_DATE> 2000-05-03
</WORK_HISTORY.WORK_END_DATE> <WORK_HISTORY.WORK_END_TIME> 23:34:39
</WORK_HISTORY.WORK_END_TIME> <WORK_HISTORY.DESCRIPTION> hello
</WORK_HISTORY.DESCRIPTION> <WORK_HISTORY.DESC_OVRFLW> _ </WORK_HISTORY.DESC_OVRFLW>
<WORK_HISTORY.MODIFY_DATETIME> 1 </WORK_HISTORY.MODIFY_DATETIME>
<WORK_HISTORY.TIME_STAMP> 2000-05-03-19.34.52.351299 </WORK_HISTORY.TIME_STAMP>
<WORK_HISTORY.ACTIVITY_ACTION_ID> _ </WORK_HISTORY.ACTIVITY_ACTION_ID> </HISTORY>
<HISTORY> <WORK_HISTORY.WORK_ID> _ </WORK_HISTORY.WORK_ID> <WORK_HISTORY.PROBLEM_ID>
00005000 </WORK_HISTORY.PROBLEM_ID> <WORK_HISTORY.USER_ID> TSD4.2GatewayA
</WORK_HISTORY.USER_ID> <WORK_HISTORY.WORK_BEGIN_DATE> 2000-05-03
</WORK_HISTORY.WORK_BEGIN_DATE> <WORK_HISTORY.WORK_BEGIN_TIME> 23:34:54
</WORK_HISTORY.WORK_BEGIN_TIME> <WORK_HISTORY.WORK_END_DATE> 2000-05-03
</WORK_HISTORY.WORK_END_DATE> <WORK_HISTORY.WORK_END_TIME> 23:34:54
</WORK_HISTORY.WORK_END_TIME> <WORK_HISTORY.DESCRIPTION> there
</WORK_HISTORY.DESCRIPTION> <WORK_HISTORY.DESC_OVRFLW> _ </WORK_HISTORY.DESC_OVRFLW>
<WORK_HISTORY.MODIFY_DATETIME> 1 </WORK_HISTORY.MODIFY_DATETIME>
<WORK_HISTORY.TIME_STAMP> 2000-05-03-19.35.02.504746 </WORK_HISTORY.TIME_STAMP>
<WORK_HISTORY.ACTIVITY_ACTION_ID> _ </WORK_HISTORY.ACTIVITY_ACTION_ID> </HISTORY>
<HISTORY> <WORK_HISTORY.WORK_ID> _ </WORK_HISTORY.WORK_ID> <WORK_HISTORY.PROBLEM_ID>
00005000 </WORK_HISTORY.PROBLEM_ID> <WORK_HISTORY.USER_ID> TSD4.2GatewayA
</WORK_HISTORY.USER_ID> <WORK_HISTORY.WORK_BEGIN_DATE> 2000-05-03
</WORK_HISTORY.WORK_BEGIN_DATE> <WORK_HISTORY.WORK_BEGIN_TIME> 23:33:52
</WORK_HISTORY.WORK_BEGIN_TIME> <WORK_HISTORY.WORK_END_DATE> 2000-05-03
</WORK_HISTORY.WORK_END_DATE> <WORK_HISTORY.WORK_END_TIME> 23:33:52
</WORK_HISTORY.WORK_END_TIME> <WORK_HISTORY.DESCRIPTION> TSD4.2GatewayA:
ACTIVE_SLA_.ACTIVE_SLA_ID = _: ACTIVE_SLA.TERM_ID = _: ACTIVE_SLA.REFERENCE_ID = _:
ACTIVE_SLA.RELATIVE_TO = _: ACTIVE_SLA.SLA_STATE = _: ACTIVE_SLA.BREACH_DATE = _:
ACTIVE_SLA.BREACH_TIME = _: ACTIVE_SLA.NEXT_SCHEDULE_ID = _: ACTIVE_SLA.
</WORK_HISTORY.DESCRIPTION> <WORK_HISTORY.DESC_OVRFLW> _ </WORK_HISTORY.DESC_OVRFLW>
<WORK_HISTORY.MODIFY_DATETIME> 1 </WORK_HISTORY.MODIFY_DATETIME>
<WORK_HISTORY.TIME_STAMP> _ </WORK_HISTORY.TIME_STAMP>
<WORK_HISTORY.ACTIVITY_ACTION_ID> _ </WORK_HISTORY.ACTIVITY_ACTION_ID> </HISTORY>
</TSD_INBOUND_TICKET>";
            gw.createRecordInDB(xml, "00002864", "00005000");
        }
        catch(Exception e) {
            gw.log(e.toString());
            e.printStackTrace();
        }

        /*(
        gw.log("before:"+argv[0]+"---");
        gw.log("after:"+gw.escapeSingleQuote(argv[0])+"---");
```

```
        */
        //gw.log("CREATING A TICKET");
        // gw.createTicketInTSD(null);
        // gw.log("UPDATE A TICKET");
        //gw.updateTicketInTSD(null);
    }
}
```